

1000 - Greetings from LightOJ

You are one of the most talented programmers and like to solve challenging problems. And my task is to make your life a bit complex by setting some easy looking hard brain storming problems such that you can sharpen your skills by practicing here. So, I wrote a code which shows a message like the following line:

Greetings from LightOJ

After that the code will select a random problem for you from my problems database based on your previously solved problems, your skills and your weaknesses. But while I was coding for implementing this great idea, I found that, all of my problems are scattered in **2** computers. So, I have to merge them before running my code.

Now you are given the number of problems in each of the computers, you have to find the total number of problems. You can safely assume that no problem is stored in multiple computers. So, all the problems are distinct.

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case starts with a line containing two integers **a** and **b**. **a** denotes the number of problems in the first computer and **b** denotes the number of problems in the second computer. You can safely assume that **a** and **b** will be non-negative and not greater than **10**.

Output

For each case, print the case number and the total number of problems. See the samples for exact formatting.

Sample Input	Output for Sample Input
2 1 7 9 8	Case 1: 8 Case 2: 17

1001 - Opposite Task

This problem gives you a flavor the concept of special judge. That means the judge is smart enough to verify your code even though it may print different results. In this problem you are asked to find the opposite task of the previous problem.

To be specific, I have two computers where I stored my problems. Now I know the total number of problems is **n**. And there are no duplicate problems and there can be at most **10** problems in each computer. You have to find the number of problems in each of the computers.

Since there can be multiple solutions. Any valid solution will do.

Input

Input starts with an integer **T (≤ 25)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($0 \leq n \leq 20$)** denoting the total number of problems.

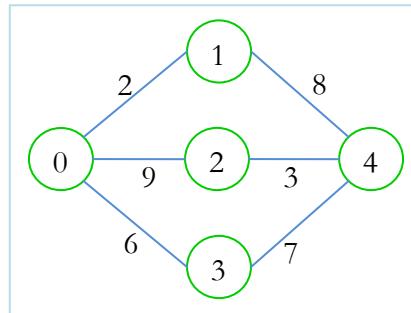
Output

For each case, print the number of problems stored in each computer in a single line. A single space should separate the non-negative integers.

Sample Input	Output for Sample Input
3 10 7 7	0 10 0 7 1 6

1002 - Country Roads

I am going to my home. There are many cities and many bi-directional roads between them. The cities are numbered from **0** to **n-1** and each road has a cost. There are **m** roads. You are given the number of my city **t** where I belong. Now from each city you have to find the minimum cost to go to my city. The cost is defined by the cost of the maximum road you have used to go to my city.



For example, in the above picture, if we want to go from 0 to 4, then we can choose

- 1) 0 - 1 - 4 which costs 8, as 8 (1 - 4) is the maximum road we used
- 2) 0 - 2 - 4 which costs 9, as 9 (0 - 2) is the maximum road we used
- 3) 0 - 3 - 4 which costs 7, as 7 (3 - 4) is the maximum road we used

So, our result is 7, as we can use 0 - 3 - 4.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a blank line and two integers **n ($1 \leq n \leq 500$)** and **m ($0 \leq m \leq 16000$)**. The next **m** lines, each will contain three integers **u, v, w ($0 \leq u, v < n, u \neq v, 1 \leq w \leq 20000$)** indicating that there is a road between **u** and **v** with cost **w**. Then there will be a single integer **t ($0 \leq t < n$)**. There can be multiple roads between two cities.

Output

For each case, print the case number first. Then for all the cities (from **0** to **n-1**) you have to print the cost. If there is no such path, print '**Impossible**'.

Sample Input	Output for Sample Input
<pre> 2 5 6 0 1 5 0 1 4 2 1 3 3 0 7 3 4 6 3 1 8 1 5 4 0 1 5 0 1 4 2 1 3 3 4 7 1 </pre>	<pre> Case 1: 4 0 3 7 7 Case 2: 4 0 3 Impossible Impossible </pre>

Note

Dataset is huge, user faster I/O methods.

1003 - Drunk

One of my friends is always drunk. So, sometimes I get a bit confused whether he is drunk or not. So, one day I was talking to him, about his drinks! He began to describe his way of drinking. So, let me share his ideas a bit. I am expressing in my words.

There are many kinds of drinks, which he used to take. But there are some rules; there are some drinks that have some pre requisites. Suppose if you want to take wine, you should have taken soda, water before it. That's why to get real drunk is not that easy.

Now given the name of some drinks! And the prerequisites of the drinks, you have to say that whether it's possible to get drunk or not. To get drunk, a person should take all the drinks.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with an integer **m ($1 \leq m \leq 10000$)**. Each of the next **m** lines will contain two names each in the format **a b**, denoting that you must have **a** before having **b**. The names will contain at most **10** characters with no blanks.

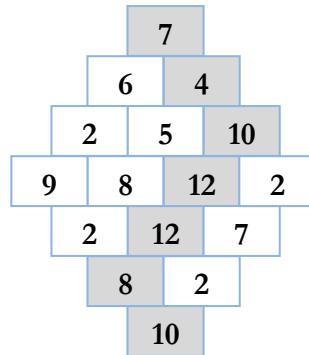
Output

For each case, print the case number and '**Yes**' or '**No**', depending on whether it's possible to get drunk or not.

Sample Input	Output for Sample Input
2 2 soda wine water wine 3 soda wine water wine wine water	Case 1: Yes Case 2: No

1004 – Monkey Banana Problem

You are in the world of mathematics to solve the great "Monkey Banana Problem". It states that, a monkey enters into a diamond shaped two dimensional array and can jump in any of the adjacent cells **down** from its current position (see figure). While moving from one cell to another, the monkey eats all the bananas kept in that cell. The monkey enters into the array from the upper part and goes out through the lower part. Find the maximum number of bananas the monkey can eat.



Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

Every case starts with an integer **N** ($1 \leq N \leq 100$). It denotes that, there will be $2*N - 1$ rows. The i^{th} ($1 \leq i \leq N$) line of next **N** lines contains exactly **i** numbers. Then there will be **N - 1** lines. The j^{th} ($1 \leq j < N$) line contains **N - j** integers. Each number is greater than zero and less than 2^{15} .

Output

For each case, print the case number and maximum number of bananas eaten by the monkey.

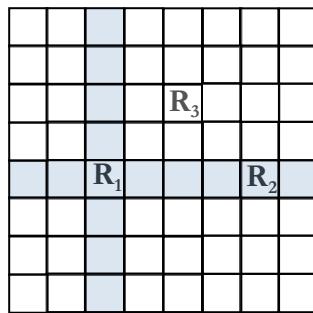
Sample Input	Output for Sample Input
2 4 7 6 4 2 5 10 9 8 12 2 2 12 7 8 2 10 2 1 2 3 1	Case 1: 63 Case 2: 5

Note

Dataset is huge, use faster I/O methods.

1005 - Rooks

A rook is a piece used in the game of chess which is played on a board of square grids. A rook can only move vertically or horizontally from its current position and two rooks attack each other if one is on the path of the other. In the following figure, the dark squares represent the reachable locations for rook R_1 from its current position. The figure also shows that the rook R_1 and R_2 are in attacking positions where R_1 and R_3 are not. R_2 and R_3 are also in non-attacking positions.



Now, given two numbers n and k , your job is to determine the number of ways one can put k rooks on an $n \times n$ chessboard so that no two of them are in attacking positions.

Input

Input starts with an integer T (≤ 350), denoting the number of test cases.

Each case contains two integers n ($1 \leq n \leq 30$) and k ($0 \leq k \leq n^2$).

Output

For each case, print the case number and total number of ways one can put the given number of rooks on a chessboard of the given size so that no two of them are in attacking positions. You may safely assume that this number will be less than 10^{17} .

Sample Input	Output for Sample Input
8	Case 1: 1
1 1	Case 2: 4
2 1	Case 3: 9
3 1	Case 4: 16
4 1	Case 5: 72
4 2	Case 6: 96
4 3	Case 7: 24
4 4	Case 8: 0
4 5	

1006 - Hex-a-bonacci

Given a code (not optimized), and necessary inputs, you have to find the output of the code for the inputs. The code is as follows:

```
int a, b, c, d, e, f;
int fn( int n ) {
    if( n == 0 ) return a;
    if( n == 1 ) return b;
    if( n == 2 ) return c;
    if( n == 3 ) return d;
    if( n == 4 ) return e;
    if( n == 5 ) return f;
    return( fn(n-1) + fn(n-2) + fn(n-3) + fn(n-4) + fn(n-5) + fn(n-6) );
}
int main() {
    int n, caseno = 0, cases;
    scanf("%d", &cases);
    while( cases-- ) {
        scanf("%d %d %d %d %d %d", &a, &b, &c, &d, &e, &f, &n);
        printf("Case %d: %d\n", ++caseno, fn(n) % 10000007);
    }
    return 0;
}
```

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains seven integers, **a, b, c, d, e, f and n**. All integers will be non-negative and **$0 \leq n \leq 10000$** and the each of the others will be fit into a 32-bit integer.

Output

For each case, print the output of the given code. The given code may have integer overflow problem in the compiler, so be careful.

Sample Input	Output for Sample Input
5 0 1 2 3 4 5 20 3 2 1 5 0 1 9 4 12 9 4 5 6 15 9 8 7 6 5 4 3 3 4 3 2 54 5 4	Case 1: 216339 Case 2: 79 Case 3: 16636 Case 4: 6 Case 5: 54

1007 - Mathematically Hard

Mathematically some problems look hard. But with the help of the computer, some problems can be easily solvable.

In this problem, you will be given two integers **a** and **b**. You have to find the summation of the scores of the numbers from **a** to **b** (inclusive). The score of **a** number is defined as the following function.

score (x) = n², where **n** is the number of relatively prime numbers with **x**, which are smaller than **x**

For example,

For 6, the relatively prime numbers with 6 are 1 and 5. So, $\text{score}(6) = 2^2 = 4$.

For 8, the relatively prime numbers with 8 are 1, 3, 5 and 7. So, $\text{score}(8) = 4^2 = 16$.

Now you have to solve this task.

Input

Input starts with an integer **T** ($\leq 10^5$), denoting the number of test cases.

Each case will contain two integers **a** and **b** ($2 \leq a \leq b \leq 5 * 10^6$).

Output

For each case, print the case number and the summation of all the scores from **a** to **b**.

Sample Input	Output for Sample Input
3 6 6 8 8 2 20	Case 1: 4 Case 2: 16 Case 3: 1237

Note

Euler's totient function $\phi(n)$ applied to a positive integer **n** is defined to be the number of positive integers less than or equal to **n** that are relatively prime to **n**. $\phi(n)$ is read "phi of n."

Given the general prime factorization of $n = p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m}$, one can compute $\phi(n)$ using the formula

$$\phi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_m}\right).$$

1008 – Fibsieve’s Fantabulous Birthday

Fibsieve had a fantabulous (yes, it's an actual word) birthday party this year. He had so many gifts that he was actually thinking of not having a party next year.

Among these gifts there was an $N \times N$ glass chessboard that had a light in each of its cells. When the board was turned on a distinct cell would light up every second, and then go dark.

The cells would light up in the sequence shown in the diagram. Each cell is marked with the second in which it would light up.

25	24	23	22	21
10	11	12	13	20
9	8	7	14	19
2	3	6	15	18
1	4	5	16	17

(The numbers in the grids stand for the time when the corresponding cell lights up)

In the first second the light at cell (1, 1) would be on. And in the 5th second the cell (3, 1) would be on. Now, Fibsieve is trying to predict which cell will light up at a certain time (given in seconds). Assume that N is large enough.

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case will contain an integer S ($1 \leq S \leq 10^{15}$) which stands for the time.

Output

For each case you have to print the case number and two numbers (x, y) , the column and the row number.

Sample Input	Output for Sample Input
3	Case 1: 2 3
8	Case 2: 5 4
20	Case 3: 1 5
25	

1009 – Back to Underworld

The Vampires and Lykans are fighting each other to death. The war has become so fierce that, none knows who will win. The humans want to know who will survive finally. But humans are afraid of going to the battlefield.

So, they made a plan. They collected the information from the newspapers of Vampires and Lykans. They found the information about all the dual fights. Dual fight means a fight between a Lykan and a Vampire. They know the name of the dual fighters, but don't know which one of them is a Vampire or a Lykan.

So, the humans listed all the rivals. They want to find the maximum possible number of Vampires or Lykans.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case contains an integer **n ($1 \leq n \leq 10^5$)**, denoting the number of dual fights. Each of the next **n** lines will contain two different integers **u v ($1 \leq u, v \leq 20000$)** denoting there was a fight between **u** and **v**. No rival will be reported more than once.

Output

For each case, print the case number and the maximum possible members of any race.

Sample Input	Output for Sample Input
2 2 1 2 2 3 3 1 2 2 3 4 2	Case 1: 2 Case 2: 3

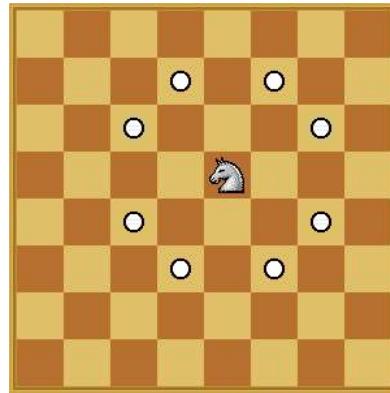
Note

Dataset is huge, use faster I/O methods.

1010 – Knights in Chessboard

Given an $m \times n$ chessboard where you want to place chess knights. You have to find the number of maximum knights that can be placed in the chessboard such that no two knights attack each other.

Those who are not familiar with chess knights, note that a chess knight can attack 8 positions in the board as shown in the picture below.



Input

Input starts with an integer T (≤ 41000), denoting the number of test cases.

Each case contains two integers m, n ($1 \leq m, n \leq 200$). Here m and n corresponds to the number of rows and the number of columns of the board respectively.

Output

For each case, print the case number and maximum number of knights that can be placed in the board considering the above restrictions.

Sample Input	Output for Sample Input
3	Case 1: 32
8 8	Case 2: 11
3 7	Case 3: 20
4 10	

1011 – Marriage Ceremonies

You work in a company which organizes marriages. Marriages are not that easy to be made, so, the job is quite hard for you.

The job gets more difficult when people come here and give their bio-data with their preference about opposite gender. Some give priorities to family background, some give priorities to education, etc.

Now your company is in a danger and you want to save your company from this financial crisis by arranging as much marriages as possible. So, you collect **N** bio-data of men and **N** bio-data of women. After analyzing quite a lot you calculated the priority index of each pair of men and women.

Finally you want to arrange **N** marriage ceremonies, such that the total priority index is maximized. Remember that each man should be paired with a woman and only monogamous families should be formed.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains an integer **N ($1 \leq n \leq 16$)**, denoting the number of men or women. Each of the next **N** lines will contain **N** integers each. The **jth** integer in the **ith** line denotes the priority index between the **ith** man and **jth** woman. All the integers will be positive and not greater than **10000**.

Output

For each case, print the case number and the maximum possible priority index after all the marriages have been arranged.

Sample Input	Output for Sample Input
2 2 1 5 2 1 3 1 2 3 6 5 4 8 1 2	Case 1: 7 Case 2: 16

1012 - Guilty Prince

Once there was a king named Akbar. He had a son named Shahjahan. For an unforgivable reason the king wanted him to leave the kingdom. Since he loved his son he decided his son would be banished in a new place. The prince became sad, but he followed his father's will. In the way he found that the place was a combination of land and water. Since he didn't know how to swim, he was only able to move on the land. He didn't know how many places might be his destination. So, he asked your help.

For simplicity, you can consider the place as a rectangular grid consisting of some cells. A cell can be a land or can contain water. Each time the prince can move to a new cell from his current position if they share a side.

Now write a program to find the number of cells (unit land) he could reach including the cell he was living.

Input

Input starts with an integer **T (≤ 500)**, denoting the number of test cases.

Each case starts with a line containing two positive integers **W** and **H**; **W** and **H** are the numbers of cells in the **x** and **y** directions, respectively. **W** and **H** are not more than 20.

There will be **H** more lines in the data set, each of which includes **W** characters. Each character represents the status of a cell as follows.

- 1) '!' - land
- 2) '#' - water
- 3) '@' - initial position of prince (appears exactly once in a dataset)

Output

For each case, print the case number and the number of cells he can reach from the initial position (including it).

Sample Input	Output for Sample Input
<pre> 4 6 9#.# #@...# .#..#. 11 9 .#..... .#.#####. .#.#....#. .#.#.#.#.#. .#.#..@#.#. .#.####.#. .#....#. .########. 11 6 ..#..#..#.. ..#..#..#.. ..#..#..### ..#..#..#@. ..#..#..#.. ..#..#..#.. 7 7 ..#.#.. ..#.#.. ###.### ...@... ###.### ..#.#.. ..#.#.. </pre>	<pre> Case 1: 45 Case 2: 59 Case 3: 6 Case 4: 13 </pre>

1013 – Love Calculator

Yes, you are developing a '**Love calculator**'. The software would be quite complex such that nobody could crack the exact behavior of the software.

So, given two names your software will generate the percentage of their '**love**' according to their names. The software requires the following things:

1. The length of the shortest string that contains the names as subsequence.
2. Total number of unique shortest strings which contain the names as subsequence.

Now your task is to find these parts.

Input

Input starts with an integer **T (≤ 125)**, denoting the number of test cases.

Each of the test cases consists of two lines each containing a name. The names will contain no more than **30** capital letters.

Output

For each of the test cases, you need to print one line of output. The output for each test case starts with the test case number, followed by the shortest length of the string and the number of unique strings that satisfies the given conditions.

You can assume that the number of unique strings will always be less than 2^{63} . Look at the sample output for the exact format.

Sample Input	Output for Sample Input
3 USA USSR LAILI MAJNU SHAHJAHAN MOMTAJ	Case 1: 5 3 Case 2: 9 40 Case 3: 13 15

1014 – Ifter Party

I have an Ifter party at the 5th day of Ramadan for the contestants. For this reason I have invited **C** contestants and arranged **P** piaju's (some kind of food, specially made for Ifter). Each contestant ate **Q** piaju's and **L** piaju's were left (**L < Q**).

Now you have to find the number of piaju's each contestant ate.

Input

Input starts with an integer **T** (≤ 325), denoting the number of test cases.

Each case contains two non-negative integers **P** and **L** ($0 \leq L < P < 2^{31}$).

Output

For each case, print the case number and the number of possible integers in ascending order. If no such integer is found print '**impossible**'.

Sample Input	Output for Sample Input
4 10 0 13 2 300 98 1000 997	Case 1: 1 2 5 10 Case 2: 11 Case 3: 101 202 Case 4: impossible

1015 – Brush (I)

Sometimes I feel angry to arrange contests, because I am too lazy. Today I am arranging a contest for AIUB students. So, I made a plan. While they will be busy with the contest, as a punishment I will cover their rooms with dusts. So, when they will be back, they will surely get angry, and it will cause them some pain.

So, at first, I will make up my mind, that means I will fix the amount of dusts for each student. This amount may not be same for all. Now you are given the amount of dust unit for each student. You have to help me finding the total dust unit I have to collect to cause them pain.

But there is a problem, my random function which generates dust units for students has a bug, it sometimes returns negative numbers. If a student gets negative number, I think he is lucky, so I will not cause him any pain with dusts.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains an integer **N ($1 \leq N \leq 1000$)**, means that there are **N** students. The next line will contain **N** integers separated by spaces which denote the dust unit for all students. The dust unit for any student will not contain more than two digits.

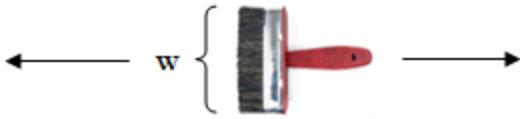
Output

For each case print the case number and the total required dust units.

Sample Input	Output for Sample Input
2 3 1 5 10 2 1 99	Case 1: 16 Case 2: 100

1016 - Brush (II)

After the long contest, Samee returned home and got angry after seeing his room dusty. Who likes to see a dusty room after a brain storming programming contest? After checking a bit he found a brush in his room which has width w .



Dusts are defined as 2D points. And since they are scattered everywhere, Samee is a bit confused what to do. So, he attached a rope with the brush such that it can be moved horizontally (in X axis) with the help of the rope but in straight line. He places it anywhere and moves it. For example, the y co-ordinate of the bottom part of the brush is 2 and its width is 3, so the y coordinate of the upper side of the brush will be 5. And if the brush is moved, all dusts whose y co-ordinates are between 2 and 5 (inclusive) will be cleaned. After cleaning all the dusts in that part, Samee places the brush in another place and uses the same procedure. He defined a **move** as placing the brush in a place and cleaning all the dusts in the horizontal zone of the brush.

You can assume that the rope is sufficiently large. Now Samee wants to clean the room with minimum number of moves. Since he already had a contest, his head is messy. So, help him.

Input

Input starts with an integer T (≤ 15), denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers N ($1 \leq N \leq 50000$) and w ($1 \leq w \leq 10000$), means that there are N dust points. Each of the next N lines will contain two integers: x_i , y_i , denoting coordinates of the dusts. You can assume that ($-10^9 \leq x_i, y_i \leq 10^9$) and all points are distinct.

Output

For each case print the case number and the minimum number of moves.

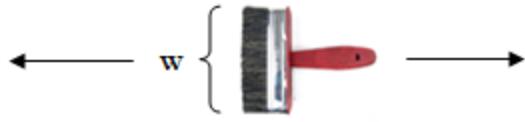
Sample Input	Output for Sample Input
2 3 2 0 0 20 2 30 2 3 1 0 0 20 2 30 2	Case 1: 1 Case 2: 2

Note

Data set is huge, so use faster I/O methods.

1017 – Brush (III)

Samir returned home from the contest and got angry after seeing his room dusty. Who likes to see a dusty room after a brain storming programming contest? After checking a bit he found a brush in his room which has width w . Dusts are defined as 2D points. And since they are scattered everywhere, Samir is a bit confused what to do. He asked Samee and found his idea. So, he attached a rope with the brush such that it can be moved horizontally (in X axis) with the help of the rope but in straight line. He places it anywhere and moves it. For example, the y co-ordinate of the bottom part of the brush is 2 and its width is 3, so the y coordinate of the upper side of the brush will be 5. And if the brush is moved, all dusts whose y co-ordinates are between 2 and 5 (inclusive) will be cleaned. After cleaning all the dusts in that part, Samir places the brush in another place and uses the same procedure. He defined a **move** as placing the brush in a place and cleaning all the dusts in the horizontal zone of the brush.



You can assume that the rope is sufficiently large. Since Samir is too lazy, he doesn't want to clean all the room. Instead of doing it he thought that he would use at most k moves. Now he wants to find the maximum number of dust units he can clean using at most k moves. Please help him.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a blank line. The next line contains three integers N ($1 \leq N \leq 100$), w ($1 \leq w \leq 10000$) and k ($1 \leq k \leq 100$). N means that there are N dust points. Each of the next N lines contains two integers: x_i y_i denoting the coordinates of the dusts. You can assume that ($-10^9 \leq x_i, y_i \leq 10^9$) and all points are distinct.

Output

For each case print the case number and the maximum number of dusts Samir can clean using at most k moves.

Sample Input	Output for Sample Input
2 3 2 1 0 0 20 2 30 2 3 1 1 0 0 20 2 30 2	Case 1: 3 Case 2: 2

1018 – Brush (IV)

Mubashwir returned home from the contest and got angry after seeing his room dusty. Who likes to see a dusty room after a brain storming programming contest? After checking a bit he found an old toothbrush in his room. Since the dusts are scattered everywhere, he is a bit confused what to do. So, he called Shakib. Shakib said that, 'Use the brush recursively and clean all the dust, I am cleaning my dust in this way!'

So, Mubashwir got a bit confused, because it's just a tooth brush. So, he will move the brush in a straight line and remove all the dust. Assume that the tooth brush only removes the dusts which lie on the line. But since he has a tooth brush so, he can move the brush in any direction. So, he counts a move as driving the tooth brush in a straight line and removing the dusts in the line.

Now he wants to find the maximum number of moves to remove all dusts. You can assume that dusts are defined as **2D** points, and if the brush touches a point, it's cleaned. Since he already had a contest, his head is messy. That's why he wants your help.

Input

Input starts with an integer **T (≤ 1000)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains three integers **N ($1 \leq N \leq 16$)**. **N** means that there are **N** dust points. Each of the next **N** lines will contain two integers **x_i, y_i** denoting the coordinate of a dust unit. You can assume that **(-1000 \leq x_i, y_i \leq 1000)** and all points are distinct.

Output

For each case print the case number and the minimum number of moves.

Sample Input	Output for Sample Input
2 3 0 0 1 1 2 2	Case 1: 1 Case 2: 2
3 0 0 1 1 2 3	

1019 - Brush (V)

Tanvir returned home from the contest and got angry after seeing his room dusty. Who likes to see a dusty room after a brain storming programming contest? After checking a bit he found that there is no brush in him room. So, he called Atiq to get a brush. But as usual Atiq refused to come. So, Tanvir decided to go to Atiq's house.

The city they live in is divided by some junctions. The junctions are connected by two way roads. They live in different junctions. And they can go to one junction to other by using the roads only.

Now you are given the map of the city and the distances of the roads. You have to find the minimum distance Tanvir has to travel to reach Atiq's house.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers **N** ($2 \leq N \leq 100$) and **M** ($0 \leq M \leq 1000$), means that there are **N** junctions and **M** two way roads. Each of the next **M** lines will contain three integers **u v w** ($1 \leq u, v \leq N, w \leq 1000$), it means that there is a road between junction **u** and **v** and the distance is **w**. You can assume that Tanvir lives in the 1st junction and Atiq lives in the **Nth** junction. There can be multiple roads between same pair of junctions.

Output

For each case print the case number and the minimum distance Tanvir has to travel to reach Atiq's house. If it's impossible, then print '**Impossible**'.

Sample Input	Output for Sample Input
2 3 2 1 2 50 2 3 10 3 1 1 2 40	Case 1: 60 Case 2: Impossible

1020 – A Childhood Game

Alice and Bob are playing a game with marbles; you may have played this game in childhood. The game is played by alternating turns. In each turn a player can take exactly one or two marbles.

Both Alice and Bob know the number of marbles initially. Now the game can be started by any one. But the winning condition depends on the player who starts it. If Alice starts first, then the player who takes the last marble loses the game. If Bob starts first, then the player who takes the last marble wins the game.

Now you are given the initial number of marbles and the name of the player who starts first. Then you have to find the winner of the game if both of them play optimally.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case contains an integer n ($1 \leq n < 2^{31}$) and the name of the player who starts first.

Output

For each case, print the case number and the name of the winning player.

Sample Input	Output for Sample Input
3 1 Alice 2 Alice 3 Bob	Case 1: Bob Case 2: Alice Case 3: Alice

1021 – Painful Bases

As you know that sometimes base conversion is a painful task. But still there are interesting facts in bases.

For convenience let's assume that we are dealing with the bases from 2 to 16. The valid symbols are **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F**. And you can assume that all the numbers given in this problem are valid. For example **67AB** is not a valid number of base **11**, since the allowed digits for base **11** are **0** to **A**.

Now in this problem you are given a base, an integer **K** and a valid number in the base which contains distinct digits. You have to find the number of permutations of the given number which are divisible by **K**. **K** is given in decimal.

For this problem, you can assume that numbers with leading zeroes are allowed. So, **096** is a valid integer.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. After that there will be two integers, **base ($2 \leq \text{base} \leq 16$)** and **K ($1 \leq K \leq 20$)**. The next line contains a valid integer in that base which contains distinct digits, that means in that number no digit occurs more than once.

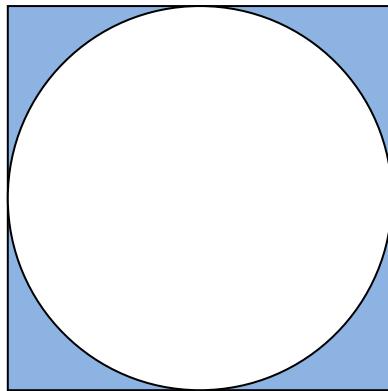
Output

For each case, print the case number and the desired result.

Sample Input	Output for Sample Input
3 2 2 10 10 2 5681 16 1 ABCDEF0123456789	Case 1: 1 Case 2: 12 Case 3: 20922789888000

1022 – Circle in Square

A circle is placed perfectly into a square. The term perfectly placed means that each side of the square is touched by the circle, but the circle doesn't have any overlapping part with the square. See the picture below.



Now you are given the radius of the circle. You have to find the area of the shaded region (blue part). Assume that $\text{pi} = 2 * \text{acos}(0.0)$ (**acos** means **cos inverse**).

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case contains a floating point number **r** ($0 < r \leq 1000$) denoting the radius of the circle. And you can assume that **r** contains at most four digits after the decimal point.

Output

For each case, print the case number and the shaded area rounded to two places after the decimal point.

Sample Input	Output for Sample Input
3	Case 1: 343.36
20	Case 2: 777.26
30.091	Case 3: 6511.05
87.0921	

Note

This problem doesn't have special judge. So, be careful about precision problems. Better to add a small value to your result to avoid precision problems. For example, add 10^{-9} to your result.

1023 - Discovering Permutations

In this problem you have to find the permutations using the first **N** English capital letters. Since there can be many permutations, you have to print the first **K** permutations.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains two integers **N, K** ($1 \leq N \leq 26, 1 \leq K \leq 30$).

Output

For each case, print the case number in a line. Then print the first **K** permutations that contain the first **N** English capital letters in alphabetical order. If there are less than **K** permutations then print all of them.

Sample Input	Output for Sample Input
2 3 8 10 10	Case 1: ABC ACB BAC BCA CAB CBA Case 2: ABCDEFGHIJ ABCDEFGHIJI ABCDEFGHIJH ABCDEFGHIJH ABCDEFGHIJHI ABCDEFGHIJH ABCDEFGHIJH ABCDEFGHIJ ABCDEFGHIJ ABCDEFGHIJG ABCDEFGHIJG

1024 - Eid

In a strange planet there are **n** races. They are completely different as well as their food habits. Each race has a food-eating period. That means the i^{th} race eats after every x_i de-sec (de-sec is the unit they use for counting time and it is used for both singular and plural). And at that particular de-sec they pass the whole day eating.

The planet declared the de-sec as 'Eid' in which all the races eat together.

Now given the eating period for every race you have to find the number of de-sec between two consecutive Eids.

Input

Input starts with an integer **T** (≤ 225), denoting the number of test cases.

Each case of input will contain an integer **n** ($2 \leq n \leq 1000$) in a single line. The next line will contain **n** integers separated by spaces. The i^{th} integer of this line will denote the eating period for the i^{th} race. These integers will be between **1** and **10000**.

Output

For each case of input you should print a line containing the case number and the number of de-sec between two consecutive Eids. Check the sample input and output for more details. The result can be big. So, use big integer calculations.

Sample Input	Output for Sample Input
2 3 2 20 10 4 5 6 30 60	Case 1: 20 Case 2: 60

1025 – The Specials Menu

Feuzem is an unemployed computer scientist who spends his days working at odd-jobs. While on the job he always manages to find algorithmic problems within mundane aspects of everyday life.

Today, while writing down the specials menu at the restaurant he's working at, he felt irritated by the lack of palindromes (strings which stay the same when reversed) on the menu. Feuzem is a big fan of palindromic problems, and started thinking about the number of ways he could remove letters from a particular word so that it would become a palindrome.

Two ways that differ due to order of removing letters are considered the same. And it can also be the case that no letters have to be removed to form a palindrome.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case contains a single word **W ($1 \leq \text{length}(W) \leq 60$)**.

Output

For each case, print the case number and the total number of ways to remove letters from W such that it becomes a palindrome.

Sample Input	Output for Sample Input
3 SALADS PASTA YUMMY	Case 1: 15 Case 2: 8 Case 3: 11

1026 - Critical Links

In a computer network a link **L**, which interconnects two servers, is considered critical if there are at least two servers **A** and **B** such that all network interconnection paths between **A** and **B** pass through **L**. Removing a critical link generates two disjoint sub-networks such that any two servers of a sub-network are interconnected. For example, the network shown in figure 1 has three critical links that are marked red: **0 - 1**, **3 - 4** and **6 - 7** in figure 2.

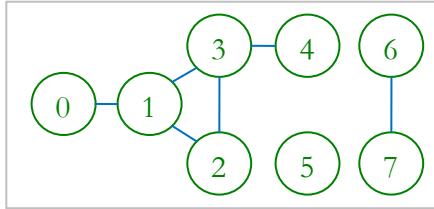


Figure 1: Original Graph

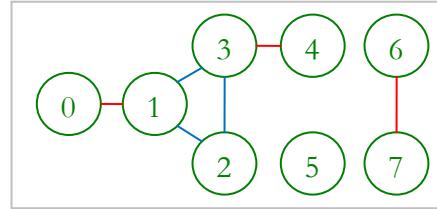


Figure 2: The Critical Links

It is known that:

1. The connection links are bi-directional.
2. A server is not directly connected to itself.
3. Two servers are interconnected if they are directly connected or if they are interconnected with the same server.
4. The network can have stand-alone sub-networks.

Write a program that finds all critical links of a given computer network.

Input

Input starts with an integer **T** (≤ 15), denoting the number of test cases.

Each case starts with a blank line. The next line will contain **n** ($0 \leq n \leq 10000$) denoting the number of nodes. Each of the next **n** lines will contain some integers in the following format:

u (k) v₁ v₂ ... v_k

Where **u** is the node identifier, **k** is the number of adjacent nodes; **v₁, v₂ ... v_k** are the adjacent nodes of **u**. You can assume that there are at most **100000** edges in total in a case. Dataset is huge, so use faster i/o methods.

Output

For each case, print the case number first. Then you should print the number of critical links and the critical links, one link per line, starting from the beginning of the line, as shown in the sample output below. The links are listed in ascending order according to their first element and then second element. Since the graph is bidirectional, print a link **u v** if **u < v**.

Sample Input	Output for Sample Input
<pre> 3 8 0 (1) 1 1 (3) 2 0 3 2 (2) 1 3 3 (3) 1 2 4 4 (1) 3 7 (1) 6 6 (1) 7 5 (0) 0 2 0 (1) 1 1 (1) 0 </pre>	<pre> Case 1: 3 critical links 0 - 1 3 - 4 6 - 7 Case 2: 0 critical links Case 3: 1 critical links 0 - 1 </pre>

Note

Dataset is huge, use faster I/O methods.

1027 – A Dangerous Maze

You are in a maze; seeing **n** doors in front of you in beginning. You can choose any door you like. The probability for choosing a door is equal for all doors.

If you choose the **ith** door, it can either take you back to the same position where you begun in **x_i** minutes, or can take you out of the maze after **x_i** minutes. If you come back to the same position, you can't remember anything. So, every time you come to the beginning position, you have no past experience.

Now you want to find the expected time to get out of the maze.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains a blank line and an integer **n ($1 \leq n \leq 100$)** denoting the number of doors. The next line contains **n** space separated integers. If the **ith** integer (**x_i**) is positive, you can assume that the **ith** door will take you out of maze after **x_i** minutes. If it's negative, then the **ith** door will take you back to the beginning position after **abs(x_i)** minutes. You can safely assume that **1 ≤ abs(x_i) ≤ 10000**.

Output

For each case, print the case number and the expected time to get out of the maze. If it's impossible to get out of the maze, print '**inf**'. Print the result in **p/q** format. Where **p** is the numerator of the result and **q** is the denominator of the result and they are relatively prime. See the samples for details.

Sample Input	Output for Sample Input
3 1 1 2 -10 -3 3 3 -6 -9	Case 1: 1/1 Case 2: inf Case 3: 18/1

1028 – Trailing Zeroes (I)

We know what a base of a number is and what the properties are. For example, we use decimal number system, where the base is **10** and we use the symbols - {**0, 1, 2, 3, 4, 5, 6, 7, 8, 9**}. But in different bases we use different symbols. For example in binary number system we use only **0** and **1**. Now in this problem, you are given an integer. You can convert it to any base you want to. But the condition is that if you convert it to any base then the number in that base should have at least one trailing zero that means a zero at the end.

For example, in decimal number system **2** doesn't have any trailing zero. But if we convert it to binary then **2** becomes **(10)₂** and it contains a trailing zero. Now you are given this task. You have to find the number of bases where the given number contains at least one trailing zero. You can use any base from two to infinite.

Input

Input starts with an integer **T (≤ 10000)**, denoting the number of test cases.

Each case contains an integer **N ($1 \leq N \leq 10^{12}$)**.

Output

For each case, print the case number and the number of possible bases where **N** contains at least one trailing zero.

Sample Input	Output for Sample Input
3 9 5 2	Case 1: 2 Case 2: 1 Case 3: 1

Note

For **9**, the possible bases are: **3** and **9**. Since in base **3**; **9** is represented as **100**, and in base **9**; **9** is represented as **10**. In both bases, **9** contains a trailing zero.

1029 – Civil and Evil Engineer

A Civil Engineer is given a task to connect **n** houses with the main electric power station directly or indirectly. The Govt has given him permission to connect exactly **n** wires to connect all of them. Each of the wires connects either two houses, or a house and the power station. The costs for connecting each of the wires are given.

Since the Civil Engineer is clever enough and tries to make some profit, he made a plan. His plan is to find the best possible connection scheme and the worst possible connection scheme. Then he will report the average of the costs.

Now you are given the task to check whether the Civil Engineer is evil or not. That's why you want to calculate the average before he reports to the Govt.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains a blank line and an integer **n** ($1 \leq n \leq 100$) denoting the number of houses. You can assume that the houses are numbered from **1** to **n** and the power station is numbered **0**. Each of the next lines will contain three integers in the form **u v w** ($0 \leq u, v \leq n, 0 < w \leq 10000, u \neq v$) meaning that you can connect **u** and **v** with a wire and the cost will be **w**. A line containing three zeroes denotes the end of the case. You may safely assume that the data is given such that it will always be possible to connect all of them. You may also assume that there will not be more than **12000** lines for a case.

Output

For each case, print the case number and the average as described. If the average is not an integer then print it in **p/q** form. Where **p** is the numerator of the result and **q** is the denominator of the result; **p** and **q** are relatively-prime. Otherwise print the integer average.

Sample Input	Output for Sample Input
3 1 0 1 10 0 1 20 0 0 0	Case 1: 15 Case 2: 229/2 Case 3: 15
3 0 1 99 0 2 10 1 2 30 2 3 30 0 0 0	
2 0 1 10 0 2 5 0 0 0	

1030 – Discovering Gold

You are in a cave, a long cave! The cave can be represented by a $1 \times N$ grid. Each cell of the cave can contain any amount of gold.

Initially you are in position **1**. Now each turn you throw a perfect **6** sided dice. If you get **X** in the dice after throwing, you add **X** to your position and collect all the gold from the new position. If your new position is outside the cave, then you keep throwing again until you get a suitable result. When you reach the N^{th} position you stop your journey. Now you are given the information about the cave, you have to find out the **expected** number of gold you can collect using the given procedure.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains a blank line and an integer **N** ($1 \leq N \leq 100$) denoting the dimension of the cave. The next line contains **N** space separated integers. The i^{th} integer of this line denotes the amount of gold you will get if you come to the i^{th} cell. You may safely assume that all the given integers will be non-negative and no integer will be greater than **1000**.

Output

For each case, print the case number and the expected number of gold you will collect. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3 1 101 2 10 3 3 3 6 9	Case 1: 101.0000000000 Case 2: 13.000 Case 3: 15

1031 – Easy Game

You are playing a two player game. Initially there are **n** integer numbers in an array and player **A** and **B** get chance to take them alternatively. Each player can take one or more numbers from the left or right end of the array but cannot take from both ends at a time. He can take as many consecutive numbers as he wants during his time. The game ends when all numbers are taken from the array by the players. The point of each player is calculated by the summation of the numbers, which he has taken. Each player tries to achieve more points from other. If both players play optimally and player **A** starts the game then how much more point can player **A** get than player **B**?

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains a blank line and an integer **N** ($1 \leq N \leq 100$) denoting the size of the array. The next line contains **N** space separated integers. You may assume that no number will contain more than 4 digits.

Output

For each test case, print the case number and the maximum difference that the first player obtained after playing this game optimally.

Sample Input	Output for Sample Input
2 4 4 -10 -20 7 4 1 2 3 4	Case 1: 7 Case 2: 10

1032 – Fast Bit Calculations

A bit is a binary digit, taking a logical value of either **1** or **0** (also referred to as "true" or "false" respectively). And every decimal number has a binary representation which is actually a series of bits. If a bit of a number is **1** and its next bit is also **1** then we can say that the number has a **1** adjacent bit. And you have to find out how many times this scenario occurs for all numbers up to **N**.

Examples:

Number	Binary	Adjacent Bits
12	1100	1
15	1111	3
27	11011	2

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case contains an integer **N** ($0 \leq N < 2^{31}$).

Output

For each test case, print the case number and the summation of all adjacent bits from **0** to **N**.

Sample Input	Output for Sample Input
7 0 6 15 20 21 22 2147483647	Case 1: 0 Case 2: 2 Case 3: 12 Case 4: 13 Case 5: 13 Case 6: 14 Case 7: 16106127360

1033 - Generating Palindromes

By definition palindrome is a string which is not changed when reversed. "**MADAM**" is a nice example of palindrome. It is an easy job to test whether a given string is a palindrome or not. But it may not be so easy to generate a palindrome.

Here we will make a palindrome generator which will take an input string and return a palindrome. You can easily verify that for a string of length **n**, no more than $(n - 1)$ characters are required to make it a palindrome. Consider "**abcd**" and its palindrome "**abcdcba**" or "**abc**" and its palindrome "**abcba**". But life is not so easy for programmers!! We always want optimal cost. And you have to find the minimum number of characters required to make a given string to a palindrome if you are only allowed to insert characters at any position of the string.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains a string of lowercase letters denoting the string for which we want to generate a palindrome. You may safely assume that the length of the string will be positive and no more than **100**.

Output

For each case, print the case number and the minimum number of characters required to make string to a palindrome.

Sample Input	Output for Sample Input
6 abcd aaaa abc aab abababaabababa pqrsabcdpqrs	Case 1: 3 Case 2: 0 Case 3: 2 Case 4: 1 Case 5: 0 Case 6: 9

1034 - Hit the Light Switches

Ronju is a night-guard at the "Lavish office buildings Ltd." headquarters. The office has a large grass field in front of the building. So every day, when Ronju comes to duty at the evening, it is his duty to turn on all the lights in the field. However, given the large size of the field and the large number of lights, it is very tiring for him to walk to each and every individual light to turn it on.

So he has devised an ingenious plan - he will swap the switches for light-sensitive triggers. A local electronic store nearby sells these funny trigger switches at a very cheap price. Once installed at a light-post, it will automatically turn that light on whenever it can sense some other light lighting up nearby. So from now on, Ronju can just manually flip a few switches, and the light from those will trigger nearby sensors, which will in turn light up some more lights nearby, and so on, gradually lighting up the whole field.

Now Ronju wonders: how many switches does he have to flip manually for this?

Input

Input starts with an integer **T (≤ 15)**, denoting the number of test cases.

Each case contains a blank line two integers **N ($1 \leq N \leq 10000$)** and **M ($0 \leq M \leq 10^5$)**, where **N** is the number of lights in the field, and **M** more lines of input follows in this input case. Each of these extra **M** lines will have two different integers **a** and **b** separated by a space, where **1 $\leq a, b \leq N$** , indicating that if the light **a** lights up, it will trigger the light **b** to turn on as well (according to their distance, brightness, sensor sensitivity, orientation and other complicated factors).

Output

For each case, print the case number and minimum number of lights that Ronju must turn on manually before all the lights in the whole field gets lit up.

Sample Input	Output for Sample Input
2 5 4 1 2 1 3 3 4 5 3 4 4 1 2 1 3 4 2 4 3	Case 1: 2 Case 2: 2

1035 – Intelligent Factorial Factorization

Given an integer N , you have to prime factorize $N!$ (factorial N).

Input

Input starts with an integer T (≤ 125), denoting the number of test cases.

Each case contains an integer N ($2 \leq N \leq 100$).

Output

For each case, print the case number and the factorization of the factorial in the following format as given in samples.

Case x: $N = p_1$ (power of p_1) * p_2 (power of p_2) * ...

Here x is the case number, $p_1, p_2 \dots$ are primes in ascending order.

Sample Input	Output for Sample Input
3	Case 1: 2 = 2 (1)
2	Case 2: 3 = 2 (1) * 3 (1)
3	Case 3: 6 = 2 (4) * 3 (2) * 5 (1)
6	

Notes

The output for the 3rd case is (if we replace space with '!') is

Case.3:.6.=.2.(4).*.3.(2).*.5.(1)

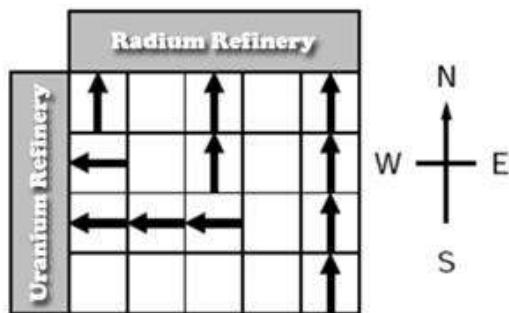
1036 – A Refining Company

Its year 2200, planet Earth is out of resources and people are relying on the resources from other planets. There are several Refining Companies who collect these resources from other planets and bring them back to Earth. The task may sound simple, but in reality it's a challenging job. The resources are scattered and after collecting them, they have to be taken to a place where they can be refined. Since some minerals are extremely dangerous, the whole process should be done very carefully. A single tiny mistake can cause a massive explosion resulting in a huge loss.

You work in such a company who collects Uranium and Radium from planet Krypton. These minerals are used for generating powers. For simplicity you have divided planet Krypton into cells that form a matrix of m rows and n columns, where the rows go from east to west and the columns go from north to south. Your advanced mine detector has detected the approximate amount of Radium and Uranium in each cell. Your company has built two refining factories, one in West and the other in North. The factory in North is used to refine Radium and the factory in West is used to refine Uranium. Your task is to design the conveyor belt system that will allow them to mine the largest amount of minerals.

There are two types of conveyor belts: the first moves minerals from east to west, the second moves minerals from south to north. In each cell you can build either type of conveyor belt, but you cannot build both of them in the same cell. If two conveyor belts of the same type are next to each other, then they can be connected. For example, the Radium mined at a cell can be transported to the Radium refinement factory via a series of south-north conveyor belts.

The minerals are very unstable, thus they have to be brought to the factories on a straight path without any turns. This means that if there is a south-north conveyor belt in a cell, but the cell north of it contains an east-west conveyor belt, then any mineral transported on the south-north conveyor belt will be lost. The minerals mined in a particular cell have to be put on a conveyor belt immediately; in the same cell (thus they cannot start the transportation in an adjacent cell). Furthermore, any Radium transported to the Uranium refinement factory will be lost, and vice versa.



Your program has to design a conveyor belt system that maximizes the total amount of minerals mined, i.e., the sum of the amount of Radium transported to the Radium refinery and the amount of Uranium to the Uranium refinery.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case begins with a blank line and two integers: **m** - the number of rows, and **n** - the number columns ($1 \leq m, n \leq 500$). The next **m** lines describe the amount of Uranium that can be found in the cells. Each of these **m** lines contains **n** integers. The first line corresponds to the northernmost row; the first integer of each line corresponds to the westernmost cell of the row. The integers are between **0** and **1000**. The next **m** lines describe in a similar fashion the amount of Radium found in the cells. Data set is huge, so use faster i/o methods.

Output

For each case of input you have to print the case number and the maximum amount of minerals you can collect.

Sample Input	Output for Sample Input
2 4 4 0 0 10 9 1 3 10 0 4 2 1 3 1 1 20 0 10 0 0 0 1 1 1 30 0 0 5 5 5 10 10 10 2 3 5 10 34 0 0 0 0 0 0 50 0 0	Case 1: 98 Case 2: 50

Note

Dataset is huge. Use faster I/O methods.

1037 – Agent 47

Agent 47 is in a dangerous Mission "Black Monster Defeat - 15". It is a secret mission and so 47 has a limited supply of weapons. As a matter of fact he has only one weapon the old weak "**KM .45 Tactical (USP)**". The mission sounds simple - "he will encounter at most 15 Targets and he has to kill them all". The main difficulty is the weapon. After huge calculations, he found a way out. That is after defeating a target, he can use target's weapon to kill other targets. So there must be an order of killing the targets so that the total number of weapon shots is minimized. As a personal programmer of Agent 47 you have to calculate the least number of shots that need to be fired to kill all the targets.



Agent 47

Now you are given a list indicating how much damage each weapon does to each target per shot, and you know how much health each target has. When a target's health is reduced to **0** or less, he is killed. 47 starts off only with the **KM .45 Tactical (USP)**, which does damage **1** per shot to any target. The list is represented as a **2D** matrix with the **ith** element containing **N** single digit numbers ('**0'-'9'**), denoting the damage done to targets **0, 1, 2, ..., N-1** by the weapon obtained from target **i**, and the health is represented as a series of **N** integers, with the **ith** element representing the amount of health that target has.

Given the list representing all the weapon damages, and the health each target has, you should find the least number of shots he needs to fire to kill all of the targets.

Input

Input starts with an integer **T (≤ 40)**, denoting the number of test cases.

Each case begins with a blank line and an integer **N ($1 \leq N \leq 15$)**. The next line contains **N** space separated integers between **1** and **10^6** denoting the health of the targets **0, 1, 2, ..., N-1**. Each of the next **N** lines contains **N** digits. The **jth** digit of the **ith** line denotes the damage done to target **j**, if you use the weapon of target **i** in each shot.

Output

For each case of input you have to print the case number and the least number of shots that need to be fired to kill all of the targets.

Sample Input	Output for Sample Input
2 3 10 10 10 010 100 111 3 3 5 7 030 500 007	Case 1: 30 Case 2: 12

1038 – Race to 1 Again

Rimi learned a new thing about integers, which is - any positive integer greater than **1** can be divided by its divisors. So, he is now playing with this property. He selects a number **N**. And he calls this **D**.

In each turn he randomly chooses a divisor of **D** (**1 to D**). Then he divides **D** by the number to obtain new **D**. He repeats this procedure until **D** becomes **1**. What is the expected number of moves required for **N** to become **1**.

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case begins with an integer **N** ($1 \leq N \leq 10^5$).

Output

For each case of input you have to print the case number and the expected value. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3 1 2 50	Case 1: 0 Case 2: 2.00 Case 3: 3.0333333333

1039 – A Toy Company

The toy company "Babies Toys" has hired you to help develop educational toys. The current project is a word toy that displays three letters at all times. Below each letter are two buttons that cause the letter above to change to the previous or next letter in alphabetical order. So, with one click of a button the letter 'c' can be changed to a 'b' or a 'd'. The alphabet is circular, so for example an 'a' can become a 'z' or a 'b' with one click.

In order to test the toy, you would like to know if a word can be reached from some starting word, given one or more constraints. A constraint defines a set of forbidden words that can never be displayed by the toy. Each constraint is formatted like "X X X", where each X is a string of lowercase letters. A word is defined by a constraint if the i^{th} letter of the word is contained in the i^{th} X of the constraint. For example, the constraint "If a tc" defines the words "lat", "fat", "lac" and "fac".

You will be given a string start, a string finish, and some forbidden strings. Calculate and return the minimum number of button presses required for the toy to show the word finish if the toy was originally showing the word start. Remember, the toy must never show a forbidden word. If it is impossible for the toy to ever show the desired word, return **-1**.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case begins with a blank line and two strings in two lines, start and finish both having exactly three characters each. The next line contains an integer **n ($1 \leq n \leq 50$)** denoting the number of forbidden words. Each of the next n lines will contain three strings each, separated by a single space. Each string (all the three strings) in a line will contain only distinct letters. Remember that start or finish can be forbidden. You can assume that all the characters are lowercase.

Output

For each case of input you have to print the case number and desired result.

Sample Input	Output for Sample Input
3 aab zna 8 a a a a a z a z a z a a a z z z a z z z a z z z aaa aaa 0 aab nnn 1 a a ab	Case 1: 15 Case 2: 0 Case 3: -1

1040 – Donation

A local charity is trying to gather donations of Ethernet cable. You realize that you probably have a lot of extra cable in your house, and make the decision that you will donate as much cable as you can spare.

You will be given the lengths (in meters) of cables between each pair of rooms in your house. You wish to keep only enough cable so that every pair of rooms in your house is connected by some chain of cables, of any length. The lengths are given in **n** lines, each having **n** integers, where **n** is the number of rooms in your house. The **jth** integer of **ith** line gives the length of the cable between rooms **i** and **j** in your house.

If both the **jth** integer of **ith** line and the **ith** integer of **jth** line are greater than **0**, this means that you have two cables connecting rooms **i** and **j**, and you can certainly donate at least one of them. If the **ith** integer of **ith** line is greater than **0**, this indicates unused cable in room **i**, which you can donate without affecting your home network in any way. **0** means no cable.

You are not to rearrange any cables in your house; you are only to remove unnecessary ones. Return the maximum total length of cables (in meters) that you can donate. If any pair of rooms is not initially connected by some path, return **-1**.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case begins with a blank line and an integer **n ($1 \leq n \leq 50$)** denoting the number of rooms in your house. Then there will be **n** lines, each having **n** space separated integers, denoting the lengths as described. Each length will be between **0** and **100**.

Output

For each case of input you have to print the case number and desired result.

Sample Input	Output for Sample Input
<pre> 3 2 27 26 1 52 4 0 10 10 0 0 0 1 1 0 0 0 2 0 0 0 0 4 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 </pre>	<pre> Case 1: 105 Case 2: 12 Case 3: -1 </pre>

1041 – Road Construction

There are several cities in the country, and some of them are connected by bidirectional roads. Unfortunately, some of the roads are damaged and cannot be used right now. Your goal is to rebuild enough of the damaged roads that there is a functional path between every pair of cities.

You are given the description of roads. Damaged roads are formatted as "**city₁ city₂ cost**" and non-damaged roads are formatted as "**city₁ city₂ 0**". In this notation **city₁** and **city₂** are the case-sensitive names of the two cities directly connected by that road. If the road is damaged, cost represents the price of rebuilding that road. Every city in the country will appear at least once in roads. And there can be multiple roads between same pair of cities.

Your task is to find the minimum cost of the roads that must be rebuilt to achieve your goal. If it is impossible to do so, print "**Impossible**".

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case begins with a blank line and an integer **m ($1 \leq m \leq 50$)** denoting the number of roads. Then there will be **m** lines, each containing the description of a road. No names will contain more than **50** characters. The road costs will lie in the range **[0, 1000]**.

Output

For each case of input you have to print the case number and the desired result.

Sample Input	Output for Sample Input
2 12 Dhaka Sylhet 0 Ctg Dhaka 0 Sylhet Chandpur 9 Ctg Barisal 9 Ctg Rajshahi 9 Dhaka Sylhet 9 Ctg Rajshahi 3 Sylhet Chandpur 5 Khulna Rangpur 7 Chandpur Rangpur 7 Dhaka Rajshahi 6 Dhaka Rajshahi 7	Case 1: 31 Case 2: Impossible
2 Rajshahi Khulna 4 Kushtia Bhola 1	

1042 – Secret Origins

This is the tale of Zephyr, the greatest time traveler the world will never know. Even those who are aware of Zephyr's existence know very little about her. For example, no one has any clue as to which time period she is originally from.

But we do know the story of the first time she set out to chart her own path in the time stream. Zephyr had just finished building her time machine which she named - "Dokhina Batash". She was making the final adjustments for her first trip when she noticed that a vital program was not working correctly. The program was supposed to take a number **N**, and find what Zephyr called its Onoroy value.

The Onoroy value of an integer **N** is the number of ones in its binary representation. For example, the number 13 (1101_2) has an Onoroy value of 3. Needless to say, this was an easy problem for the great mind of Zephyr. She solved it quickly, and was on her way.

You are now given a similar task. Find the first number after **N** which has the same Onoroy value as **N**.

Input

Input starts with an integer **T** (≤ 65), denoting the number of test cases.

Each case begins with an integer **N** ($1 \leq N \leq 10^9$).

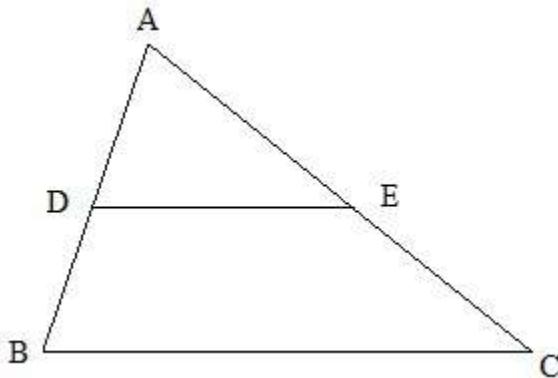
Output

For each case of input you have to print the case number and the desired result.

Sample Input	Output for Sample Input
5	Case 1: 27
23	Case 2: 14241
14232	Case 3: 395
391	Case 4: 11
7	Case 5: 16
8	

1043 – Triangle Partitioning

See the picture below.



You are given **AB**, **AC** and **BC**. **DE** is parallel to **BC**. You are also given the area ratio between **ADE** and **BDEC**. You have to find the value of **AD**.

Input

Input starts with an integer **T** (≤ 25), denoting the number of test cases.

Each case begins with four real numbers denoting **AB**, **AC**, **BC** and the ratio of **ADE** and **BDEC** (**ADE / BDEC**). You can safely assume that the given triangle is a valid triangle with positive area.

Output

For each case of input you have to print the case number and **AD**. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 100 100 100 2 10 12 14 1 7 8 9 10 8.134 9.098 7.123 5.10	Case 1: 81.6496580 Case 2: 7.07106781 Case 3: 6.6742381247 Case 4: 7.437454786

1044 - Palindrome Partitioning

A palindrome partition is the partitioning of a string such that each separate substring is a palindrome.

For example, the string "**ABACABA**" could be partitioned in several different ways, such as {"A","B","A","C","A","B","A"}, {"A","BACAB","A"}, {"ABA","C","ABA"}, or {"ABACABA"}, among others.

You are given a string **s**. Return the minimum possible number of substrings in a palindrome partition of **s**.

Input

Input starts with an integer **T (≤ 40)**, denoting the number of test cases.

Each case begins with a non-empty string **s** of uppercase letters with length no more than **1000**.

Output

For each case of input you have to print the case number and the desired result.

Sample Input	Output for Sample Input
3 AAAA ABCDEFGH QWERTYTREWQWERT	Case 1: 1 Case 2: 8 Case 3: 5

1045 - Digits of Factorial

Factorial of an integer is defined by the following function

$$\begin{aligned}f(0) &= 1 \\f(n) &= f(n - 1) * n, \text{ if}(n > 0)\end{aligned}$$

So, factorial of 5 is 120. But in different bases, the factorial may be different. For example, factorial of 5 in base 8 is 170.

In this problem, you have to find the number of digit(s) of the factorial of an integer in a certain base.

Input

Input starts with an integer **T (≤ 50000)**, denoting the number of test cases.

Each case begins with two integers **n ($0 \leq n \leq 10^6$)** and **base ($2 \leq \text{base} \leq 1000$)**. Both of these integers will be given in decimal.

Output

For each case of input you have to print the case number and the digit(s) of factorial n in the given base.

Sample Input	Output for Sample Input
5 5 10 8 10 22 3 1000000 2 0 100	Case 1: 3 Case 2: 5 Case 3: 45 Case 4: 18488885 Case 5: 1

1046 – Rider

A rider is a fantasy chess piece that can jump like a knight several times in a single move. A rider that can perform a maximum of **K** jumps during a single move is denoted as a **K**-rider. For example, a 2-rider can jump once or twice during a single move, and a 1-rider is a traditional knight.

There are some riders of different types on a chessboard. You are given a 2D board representing the layout of the pieces. The j^{th} character of the i^{th} element of board is the content of the square at row i , column j . If the character is a digit **K** between '1' and '9', the square contains a **K**-rider. Otherwise, if the character is a '.', the square is empty. Find the minimal total number of moves necessary to move all the riders to the same square. Only one piece can move during each move. Multiple riders can share the same squares all times during the process. Print -1 if it is impossible.

A traditional knight has up to 8 moves from a square with coordinates (x, y) to squares $(x+1, y+2)$, $(x+1, y-2)$, $(x+2, y+1)$, $(x+2, y-1)$, $(x-1, y+2)$, $(x-1, y-2)$, $(x-2, y+1)$, $(x-2, y-1)$, and can't move outside the chessboard.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case begins with a blank line and two integers **m**, **n** ($1 \leq m, n \leq 10$) denoting the rows and the columns of the board respectively. Each of the next **m** lines will contain **n** integers each denoting the board.

Output

For each case of input you have to print the case number the desired result.

Sample Input	Output for Sample Input
<pre> 4 3 2 .. 2. .. 3 3 1.11 10 102....2...2... 1..... ...2.1.... ...1.....21. 1 4 1..1 </pre>	<pre> Case 1: 0 Case 2: 4 Case 3: 14 Case 4: -1 </pre>

1047 - Neighbor House

The people of Mohammadpur have decided to paint each of their houses red, green, or blue. They've also decided that no two neighboring houses will be painted the same color. The neighbors of house **i** are houses **i-1** and **i+1**. The first and last houses are not neighbors.

You will be given the information of houses. Each house will contain three integers "**R G B**" (quotes for clarity only), where **R**, **G** and **B** are the costs of painting the corresponding house red, green, and blue, respectively. Return the minimal total cost required to perform the work.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case begins with a blank line and an integer **n** ($1 \leq n \leq 20$) denoting the number of houses. Each of the next **n** lines will contain 3 integers "**R G B**". These integers will lie in the range [1, 1000].

Output

For each case of input you have to print the case number and the minimal cost.

Sample Input	Output for Sample Input
2 4 13 23 12 77 36 64 44 89 76 31 78 45	Case 1: 137 Case 2: 96
3 26 40 83 49 60 57 13 89 99	

1048 - Conquering Keokradong

This winter we are going on a trip to Bandorban. The main target is to climb up to the top of Keokradong. So, we will use a trail. The trail is a continuous marked footpath that goes from Bandorban to Keokradong.

Part of the experience is also the route planning of the trip. We have a list of all possible campsites that we can use along the way and we want to do this trip so that we only stop **K** nights to camp. We also know in advance the distance between consecutive campsites and we are only allowed to camp at a campsite. Our goal is to plan the trip so that we minimize the maximum amount of walking done in a single day. In other words, if our trip involves 2 nights (3 days of walking), and we walk 9, 10, 5 miles on each day respectively, the cost (maximum amount of walking done in one day) is 10. Another schedule that involves walking 9, 6, 9 miles on each day has cost 9.

Given the distances between **N** consecutive campsites of a trail and given the number of nights for your trip, **K**, your task is to devise a camping strategy for the specified trail such that it minimizes the maximum amount of walking done in a single day. Note that the first distance value given is the distance from our start-point of the trail to our 1st campsite, and the last distance value given is the distance from our **Nth** campsite to our end-point of the trail.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains of two integers, the number of campsites, **N** ($1 \leq N \leq 1000$) and the number of nights of the trip, **K** ($1 \leq K \leq \min(N, 300)$). The following **N + 1** lines indicate the distance in miles between consecutive campsite locations. All the integers will be positive and less than **10000**.

Output

For each case of input you have to print the case number and the minimized cost as described above. Then print **K+1** lines, each containing the amount of distance covered in **ith** day. As there can be many solutions, the primary target is to find the one which ensures that each day we have to walk some distance. For ties, print the one where the distance covered in first day is maximum, then the distance covered in second day is maximum and so on.

Sample Input	Output for Sample Input
1 4 3 7 2 6 4 5	Case 1: 8 7 8 4 5

1049 – One Way Roads

Nowadays the one-way traffic is introduced all over the world in order to improve driving safety and reduce traffic jams. The government of Dhaka Division decided to keep up with new trends. Formerly all n cities of Dhaka were connected by n two-way roads in the ring, i.e. each city was connected directly to exactly two other cities, and from each city it was possible to get to any other city. Government of Dhaka introduced one-way traffic on all n roads, but it soon became clear that it's impossible to get from some of the cities to some others. Now for each road is known in which direction the traffic is directed at it, and the cost of redirecting the traffic. What is the smallest amount of money the government should spend on the redirecting of roads so that from every city you can get to any other?

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case starts with a blank line and an integer n ($3 \leq n \leq 100$) denoting the number of cities (and roads). Next n lines contain description of roads. Each road is described by three integers a_i, b_i, c_i ($1 \leq a_i, b_i \leq n, a_i \neq b_i, 1 \leq c_i \leq 100$) - road is directed from city a_i to city b_i , redirecting the traffic costs c_i .

Output

For each case of input you have to print the case number and the smallest amount of money the government should spend on the redirecting of roads so that from every city you can get to any other.

Sample Input	Output for Sample Input
4	Case 1: 1 Case 2: 2 Case 3: 39 Case 4: 0
3	
1 3 1	
1 2 1	
3 2 1	
3	
1 3 1	
1 2 5	
3 2 1	
6	
1 5 4	
5 3 8	
2 4 15	
1 6 16	
2 3 23	
4 6 42	
4	
1 2 9	
2 3 8	
3 4 7	
4 1 5	

1050 - Marbles

Your friend Jim has challenged you to a game. He has a bag containing red and blue marbles. There will be an odd number of marbles in the bag, and you go first. On your turn, you reach into the bag and remove a random marble from the bag; each marble may be selected with equal probability. After your turn is over, Jim will reach into the bag and remove a blue marble; if there is no blue marble for Jim to remove, then he wins. If the final marble removed from the bag is blue (by you or Jim), you will win. Otherwise, Jim wins.

Given the number of red and blue marbles in the bag, determine the probability that you win the game.

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case begins with two integers **R** and **B** denoting the number of red and blue marbles respectively. You can assume that $0 \leq R, B \leq 500$ and $R+B$ is odd.

Output

For each case of input you have to print the case number and your winning probability. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
5 1 2 2 3 2 5 11 6 4 11	Case 1: 0.333333333 Case 2: 0.13333333 Case 3: 0.2285714286 Case 4: 0 Case 5: 0.1218337218

1051 – Good or Bad

A string is called bad if it has 3 vowels in a row, or 5 consonants in a row, or both. A string is called good if it is not bad. You are given a string s , consisting of uppercase letters ('A'-'Z') and question marks ('?'). Return "BAD" if the string is definitely bad (that means you cannot substitute letters for question marks so that the string becomes good), "GOOD" if the string is definitely good, and "MIXED" if it can be either bad or good.

The letters 'A', 'E', 'I', 'O', 'U' are vowels, and all others are consonants.

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case begins with a non-empty string with length no more than **50**.

Output

For each case of input you have to print the case number and the result according to the description.

Sample Input	Output for Sample Input
5 FFFF?EE HELLOWORLD ABCDEFGHIJKLMNPQRSTUVWXYZ HELLO?ORLD AAA	Case 1: BAD Case 2: GOOD Case 3: BAD Case 4: MIXED Case 5: BAD

1052 – String Growth

Zibon just started his courses in Computer science. After having some lectures on programming courses he fell in love with strings. He started to play with strings and experiments on them. One day he started a string of arbitrary (of course positive) length consisting of only {**a**, **b**}. He considered it as 1st string and generated subsequent strings from it by replacing all the **b**'s with **ab** and all the **a**'s with **b**. For example, if he **i**th string is **abab**, (**i+1**)th string will be **b(ab)b(ab) = babbab**. He found that the **N**th string has length **X** and **M**th string has length **Y**. He wondered what will be length of the **K**th string. Can you help him?

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case begins with five integers **N, X, M, Y, K**. ($0 < N, M, X, Y, K < 10^9$ and $N \neq M$).

Output

For each case print one line containing the case number and **L** which is the desired length (**mod 1000000007**) or the string "**Impossible**" if it's not possible.

Sample Input	Output for Sample Input
2 3 16 5 42 6 5 1 6 10 9	Case 1: 68 Case 2: Impossible

1053 – Higher Math

You are building a house. You'd prefer if all the walls have a precise right angle relative to the ground, but you have no device to measure angles. A friend says he has a great idea how you could ensure that all walls are upright: All you need to do is step away a few feet from the wall, measure how far away you are from the wall, measure the height of the wall, and the distance from the upper edge of the wall to where you stand. Your friend tells you to do these measurements for all walls, then he'll tell you how to proceed. Sadly, just as you are done, a timber falls on your friend, and an ambulance brings him to the hospital. This is too bad, because now you have to figure out what to do with your measurements yourself.

Given the three sides of a triangle, determine if the triangle is a right triangle, i.e. if one of the triangle's angles is **90** degrees.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each test case consists of three integers **1 ≤ a, b, c ≤ 40000** separated by a space. The three integers are the lengths of the sides of a triangle.

Output

For each case, print the case number and "yes" or "no" depending on whether it's a right angle or not.

Sample Input	Output for Sample Input
2 36 77 85 40 55 69	Case 1: yes Case 2: no

1054 - Efficient Pseudo Code

Sometimes it's quite useful to write pseudo codes for problems. Actually you can write the necessary steps to solve a particular problem. In this problem you are given a pseudo code to solve a problem and you have to implement the pseudo code efficiently. Simple! Isn't it? :)

```
pseudo code
{
    take two integers n and m
    let p = n ^ m (n to the power m)
    let sum = summation of all the divisors of p
    let result = sum MODULO 1000,000,007
}
```

Now given **n** and **m** you have to find the desired result from the pseudo code. For example if **n = 12** and **m = 2**. Then if we follow the pseudo code, we get

```
pseudo code
{
    take two integers n and m
    so, n = 12 and m = 2
    let p = n ^ m (n to the power m)
    so, p = 144
    let sum = summation of all the divisors of p
    so, sum = 403, since the divisors of p are 1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 36, 48, 72, 144
    let result = sum MODULO 1000,000,007
    so, result = 403
}
```

Input

Input starts with an integer **T (≤ 5000)**, denoting the number of test cases.

Each test case will contain two integers, **n ($1 \leq n$)** and **m ($0 \leq m$)**. Each of **n** and **m** will be fit into a **32** bit signed integer.

Output

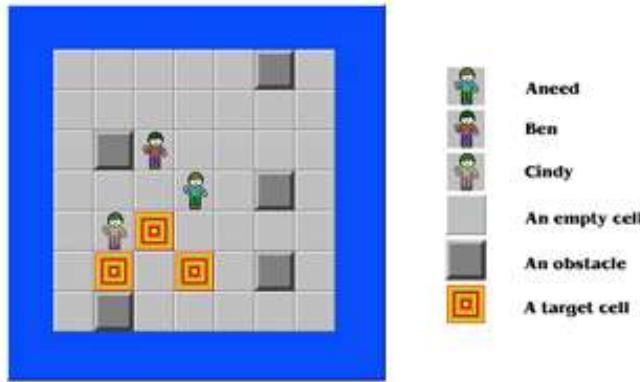
For each case of input you have to print the case number and the result according to the pseudo code.

Sample Input	Output for Sample Input
3 12 2 12 1 36 2	Case 1: 403 Case 2: 28 Case 3: 3751

1055 – Going Together

You are playing a computer game in which three robots (Aneed, Ben and Cindy) are trapped in a labyrinth. Initially all three are situated in three different locations in the maze. There are three outlets through which the robots have to exit. As expected, there are several obstacles in the maze and the robots can't go through them.

The maze can be modeled as a square grid with $N \times N$ cells. The robots are placed on three different cells into the maze. You can command them to move. A single command will be activated for the three robots simultaneously. A robot will move to a new position (whatever its current situation is) if it is an empty cell within the maze or it is one of the free target cells, otherwise the command will be ignored for that robot. Your task is to command them such a way that all of them are on three exit cells (in any order).



A move consists of one of the following (Each move takes 1 unit of time):

- Move North** The robots move one cell north.
- Move East** The robots move one cell east.
- Move South** The robots move one cell south.
- Move West** The robots move one cell west.

Each cell consists of one of the following characters:

- A** – Initial position of Aneed
- B** – Initial position of Ben
- C** – Initial position of Cindy
- .** – An empty cell
- #** - An obstacle
- X** – A target cell

You can assume that for every maze each of the letters (**A B C**) will appear exactly once and the letter **X** will appear exactly three times.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

Each case starts with an integer **N** ($2 < N < 10$). Each of the next **N** lines contains **N** characters that fill up the maze.

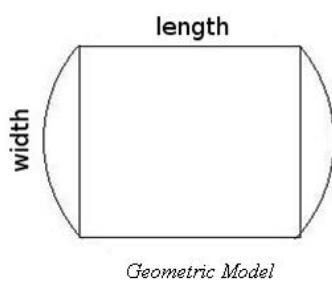
Output

For each case, output the case number followed by the minimum time required. If it is impossible to move them as described, print ‘trapped’ instead of the time.

Sample Input	Output for Sample Input
3 7#. #B.... . .A. #. . CX.... . X.X.#. . #..... 3 ABC ... XXX 3 ABC ### XXX	Case 1: 2 Case 2: 2 Case 3: trapped

1056 – Olympics

The next Olympic is approaching very shortly. It's a hard job for the organizers. There are so many things to do - preparing the venues, building the Olympic village for accommodating athletes and officials, improving the transportation of the entire city as the venues are located all over the city and also there will be great number of tourists/spectators during the Olympics.



Geometric Model



Actual View

One of the most important tasks is to build the stadium. You are appointed as a programmer to help things out in certain matters - more specifically in designing and building the athletics tracks. After some study, you find out that athletics tracks have a general shape of a rectangle with two sliced circles on two ends. Now the turf that is placed inside this rectangle is prepared elsewhere and comes in different shapes - different length to width ratios. You know one thing for certain - your track should have a perimeter of 400 meters. That's the standard length for athletics tracks. You are supplied with the design parameter - length to width ratio. You are also told that the sliced circles will be such that they are part of the same circle. You have to find the length and width of the rectangle.

Input

Input starts with an integer **T (≤ 1000)**, denoting the number of test cases.

Each case starts with the ratio of the length and width of the rectangle in the format – "**a : b**". Here, **a** and **b** will be integers and both will be between **1** and **1000** (inclusive).

Output

For each case, print the case number, the length and the width. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2	Case 1: 117.1858168 78.12387792
3 : 2	Case 2: 107.29095604 85.8327648
5 : 4	

1057 – Collecting Gold

Finally you found the city of Gold. As you are fond of gold, you start collecting them. But there are so much gold that you are getting tired collecting them.

So, you want to find the minimum effort to collect all the gold.

You can describe the city as a **2D** grid, where your initial position is marked by an '**x**'. An empty place will be denoted by a '**.**'. And the cells which contain gold will be denoted by '**g**'. In each move you can go to all **8** adjacent places inside the city.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case will start with a blank line and two integers, **m** and **n** ($0 < m, n < 20$) denoting the row and columns of the city respectively. Each of the next **m** lines will contain **n** characters describing the city. There will be exactly one '**x**' in the city and at most **15** gold positions.

Output

For each case of input you have to print the case number and the minimum steps you have to take to collect all the gold and go back to '**x**'.

Sample Input	Output for Sample Input
2 5 5 x.... g.... g.... g.... 5 5 x.... g.... g....	Case 1: 8 Case 2: 4

1058 – Parallelogram Counting

There are **n** distinct points in the plane, given by their integer coordinates. Find the number of parallelograms whose vertices lie on these points. In other words, find the number of 4-element subsets of these points that can be written as {**A**, **B**, **C**, **D**} such that **AB** || **CD**, and **BC** || **AD**. No four points are in a straight line.

Input

Input starts with an integer **T** (≤ 15), denoting the number of test cases.

The first line of each test case contains an integer **n** ($1 \leq n \leq 1000$). Each of the next **n** lines, contains 2 space-separated integers **x** and **y** (the coordinates of a point) with magnitude (absolute value) of no more than **1000000000**.

Output

For each case, print the case number and the number of parallelograms that can be formed.

Sample Input	Output for Sample Input
2 6 0 0 2 0 4 0 1 1 3 1 5 1 7 -2 -1 8 9 5 7 1 1 4 8 2 0 9 8	Case 1: 5 Case 2: 6

1059 – Air Ports

The government of a certain developing nation wants to improve transportation in one of its most inaccessible areas, in an attempt to attract investment. The region consists of several important locations that must have access to an airport.



Of course, one option is to build an airport in each of these places, but it may turn out to be cheaper to build fewer airports and have roads link them to all of the other locations. Since these are long distance roads connecting major locations in the country (e.g. cities, large villages, industrial areas), all roads are two-way. Also, there may be more than one direct road possible between two areas. This is because there may be several ways to link two areas (e.g. one road tunnels through a mountain while the other goes around it etc.) with possibly differing costs.

A location is considered to have access to an airport either if it contains an airport or if it is possible to travel by road to another location from there that has an airport.

You are given the cost of building an airport and a list of possible roads between pairs of locations and their corresponding costs. The government now needs your help to decide on the cheapest way of ensuring that every location has access to an airport. The aim is to make airport access as easy as possible, so if there are several ways of getting the minimal cost, choose the one that has the most airports.

Note: The input file is large; make sure your I/O code is fast.

Input

Input starts with an integer **T** (≤ 15), denoting the number of test cases.

Each case starts with three integers **N**, **M** and **A** ($0 < N \leq 10000$, $0 \leq M \leq 100000$, $0 < A \leq 10000$) separated by white space. **N** is the number of locations, **M** is the number of possible roads that can be built, and **A** is the cost of building an airport.

The following **M** lines each contain three integers **X**, **Y** and **C** ($1 \leq X, Y \leq N$, $0 < C \leq 10000$), separated by white space. **X** and **Y** are two locations, and **C** is the cost of building a road between **X** and **Y**.

Output

For each case, print the case number and $Y\ Z$, where Y is the minimum cost of making roads and airports so that all locations have access to at least one airport, and Z is the number of airports to be built. As mentioned earlier, if there are several answers with minimal cost, choose the one that maximizes the number of airports.

Sample Input	Output for Sample Input
2 4 4 100 1 2 10 4 3 12 4 1 41 2 3 23 5 3 1000 1 2 20 4 5 40 3 2 30	Case 1: 145 1 Case 2: 2090 2

1060 - nth Permutation

Given a string of characters, we can permute the individual characters to make new strings. At first we order the string into alphabetical order. Then we start permuting it.

For example the string '**abba**' gives rise to the following 6 distinct permutations in alphabetical order.

aabb 1
abab 2
abba 3
baab 4
baba 5
bbaa 6

Given a string, you have to find the **nth** permutation for that string. For the above case '**aabb**' is the 1st and '**baab**' is the 4th permutation.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains a non empty string of lowercase letters with length no more than **20** and an integer **n** ($0 < n < 2^{31}$).

Output

For each case, output the case number and the **nth** permutation. If the **nth** permutation doesn't exist print '**Impossible**'.

Sample Input	Output for Sample Input
3 aab 1 aab 6 aab 7	Case 1: aabb Case 2: bbaa Case 3: Impossible

1061 - N Queen Again

Given an **8 x 8** chess board where **8** queens are placed in any arbitrary order. You want to move the queens such that no one attacks each other.

In each move you can move a queen as in normal chess. That means you can move a queen vertically, horizontally or diagonally. And you can move it to multiple cells at a time but the direction can't be changed and you can't jump over any other queen. Now you want to find the minimum number of moves to do this.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case contains a blank line and an **8 x 8** board consisting of '.' and exactly eight 'q'. '.' stands for empty position and 'q' stands for a queen.

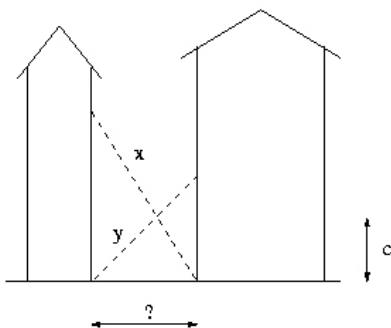
Output

For each case, output the case number and minimum number of queen moves you have to do such that no queen attacks another.

Sample Input	Output for Sample Input
2 q..... .q..... ..q..... ...q....q...q..q.q qqq..q.. .q...q.. ...q... ..q.....	Case 1: 7 Case 2: 5

1062 - Crossed Ladders

A narrow street is lined with tall buildings. An x foot long ladder is rested at the base of the building on the right side of the street and leans on the building on the left side. A y foot long ladder is rested at the base of the building on the left side of the street and leans on the building on the right side. The point where the two ladders cross is exactly c feet from the ground. How wide is the street?



Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each test case contains three positive floating point numbers giving the values of x , y , and c .

Output

For each case, output the case number and the width of the street in feet. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 30 40 10 12.619429 8.163332 3 10 10 3 10 10 1	Case 1: 26.0328775442 Case 2: 6.99999923 Case 3: 8 Case 4: 9.797958971

1063 – Ant Hills

After many years of peace, an ant-war has broken out.

In the days leading up to the outbreak of war, the ant government devoted a great deal of resources toward gathering intelligence on ant hills. It discovered the following:

1. The ant empire has a large network of ant-hills connected by bidirectional tracks.
2. It is possible to send a message from any ant hill to any other ant hill.

Now you want to stop the war. Since they sometimes attack your house and disturb you quite a lot. So, you have made a plan. You have a gun which can destroy exactly one ant-hill. So, you want to hit an ant hill if it can stop at least two other ant hills passing messages between them. Now you want the total number of ant hills you may choose to fire.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each test case contains a blank line and two integers **n ($1 \leq n \leq 10000$)**, **m ($1 \leq m \leq 20000$)**. **n** denotes the number of ant hills and **m** denotes the number of bi-directional tracks. Each of the next **m** lines will contain two different integers **a b ($1 \leq a, b \leq n$)** denoting that there is a track between **a** and **b**.

Output

For each case, print the case number and the total number of ant hills you may choose to fire.

Sample Input	Output for Sample Input
2 5 4 2 1 1 3 5 4 4 1 3 3 1 2 2 3 1 3	Case 1: 2 Case 2: 0

1064 - Throwing Dice

n common cubic dice are thrown. What is the probability that the sum of all thrown dice is at least **x**?

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each test case contains two integers **n** ($1 \leq n < 25$) and **x** ($0 \leq x < 150$). The meanings of **n** and **x** are given in the problem statement.

Output

For each case, output the case number and the probability in '**p/q**' form where **p** and **q** are relatively prime. If **q** equals 1 then print **p** only.

Sample Input	Output for Sample Input
7	Case 1: 20/27
3 9	Case 2: 0
1 7	Case 3: 1
24 24	Case 4: 11703055/78364164096
15 76	Case 5: 25/4738381338321616896
24 143	Case 6: 1/2
23 81	Case 7: 55/46656
7 38	

1065 – Number Sequence

Let's define another number sequence, given by the following function:

$$\begin{aligned}f(0) &= a \\f(1) &= b \\f(n) &= f(n-1) + f(n-2), n > 1\end{aligned}$$

When $a = 0$ and $b = 1$, this sequence gives the Fibonacci sequence. Changing the values of a and b , you can get many different sequences. Given the values of a , b , you have to find the last m digits of $f(n)$.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each test case consists of a single line containing four integers a b n m . The values of a and b range in $[0, 100]$, value of n ranges in $[0, 10^9]$ and value of m ranges in $[1, 4]$.

Output

For each case, print the case number and the last m digits of $f(n)$. However, do **NOT** print any leading zero.

Sample Input	Output for Sample Input
4 0 1 11 3 0 1 42 4 0 1 22 4 0 1 21 4	Case 1: 89 Case 2: 4296 Case 3: 7711 Case 4: 946

1066 – Gathering Food

Winter is approaching! The weather is getting colder and days are becoming shorter. The animals take different measures to adjust themselves during this season.

- Some of them "migrate." This means they travel to other places where the weather is warmer.
- Few animals remain and stay active in the winter.
- Some animals "hibernate" for all of the winter. This is a very deep sleep. The animal's body temperature drops, and its heartbeat and breathing slow down. In the fall, these animals get ready for winter by eating extra food and storing it as body fat.

For this problem, we are interested in the 3rd example and we will be focusing on 'Yogi Bear'.

Yogi Bear is in the middle of some forest. The forest can be modeled as a square grid of size $N \times N$. Each cell of the grid consists of one of the following.

- . an empty space
- # an obstacle
- [A-Z] an English alphabet

There will be at least 1 alphabet and all the letters in the grid will be distinct. If there are k letters, then it will be from the first k alphabets. Suppose $k = 3$, that means there will be exactly one **A**, one **B** and one **C**.

The letters actually represent foods lying on the ground. Yogi starts from position '**A**' and sets off with a basket in the hope of collecting all other foods. Yogi can move to a cell if it shares an edge with the current one. For some superstitious reason, Yogi decides to collect all the foods in order. That is, he first collects **A**, then **B**, then **C** and so on until he reaches the food with the highest alphabet value. Another philosophy he follows is that if he lands on a particular food he must collect it.

Help Yogi to collect all the foods in minimum number of moves so that he can have a long sleep in the winter.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains a blank line and an integer **N** ($0 < N < 11$), the size of the grid. Each of the next **N** lines contains **N** characters each.

Output

For each case, output the case number first. If it's impossible to collect all the foods, output '**Impossible**'. Otherwise, print the shortest distance.

Sample Input	Output for Sample Input
4 5 A.... ###. .B.. .#### C.DE.	Case 1: 15 Case 2: 3 Case 3: Impossible Case 4: Impossible
2 AC .B	
2 A# #B	
3 A.C ##. B..	

1067 – Combinations

Given **n** different objects, you want to take **k** of them. How many ways to can do it?

For example, say there are 4 items; you want to take 2 of them. So, you can do it 6 ways.

Take 1, 2
Take 1, 3
Take 1, 4
Take 2, 3
Take 2, 4
Take 3, 4

Input

Input starts with an integer **T** (≤ 2000), denoting the number of test cases.

Each test case contains two integers **n** ($1 \leq n \leq 10^6$), **k** ($0 \leq k \leq n$).

Output

For each case, output the case number and the desired value. Since the result can be very large, you have to print the result modulo **1000003**.

Sample Input	Output for Sample Input
3 4 2 5 0 6 4	Case 1: 6 Case 2: 1 Case 3: 15

1068 - Investigation

An integer is divisible by 3 if the sum of its digits is also divisible by 3. For example, 3702 is divisible by 3 and 12 ($3+7+0+2$) is also divisible by 3. This property also holds for the integer 9.

In this problem, we will investigate this property for other integers.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains three positive integers **A**, **B** and **K** ($1 \leq A \leq B < 2^{31}$ and $0 < K < 10000$).

Output

For each case, output the case number and the number of integers in the range **[A, B]** which are divisible by **K** and the sum of its digits is also divisible by **K**.

Sample Input	Output for Sample Input
3 1 20 1 1 20 2 1 1000 4	Case 1: 20 Case 2: 5 Case 3: 64

1069 - Lift

All of you must have noticed that the lift of AIUB is not available for students. But since you deny obeying usual rules, you always use this lift no matter what happens!

Now one day you were sleeping in the class and when you woke up you found none in the department except the guard who was in a deep sleep in his room. But luckily you found the lift on. So, you want to go to the ground floor using the lift. But the lift can be in a different floor. Then you must wait for the lift to come to your floor. Assume that it takes 4 seconds for the lift to go from any floor to its adjacent floor (up or down). It takes 3 seconds to open or close the door of the lift and you need 5 seconds to enter or exit the lift. Given your position and the lift's position you have to calculate the time for you to reach the ground floor (floor 0). Reaching ground floor means you must get out of the lift in ground floor.

Input

Input starts with an integer **T** (≤ 25), denoting the number of test cases.

Each case contains two integers. The first integer means your position (different than 0) and the second integer means the position of the lift. Assume that the department has **100** floors (may be one day it will be :D).

Output

For each case, print the case number and the calculated time in seconds.

Sample Input	Output for Sample Input
3 1 2 3 10 5 5	Case 1: 27 Case 2: 59 Case 3: 39

1070 – Algebraic Problem

Given the value of $a+b$ and ab you will have to find the value of a^n+b^n . a and b not necessarily have to be real numbers.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case contains three non-negative integers, p , q and n . Here p denotes the value of $a+b$ and q denotes the value of ab . Each number in the input file fits in a signed 32-bit integer. There will be no such input so that you have to find the value of 0^0 .

Output

For each test case, print the case number and (a^n+b^n) modulo 2^{64} .

Sample Input	Output for Sample Input
2 10 16 2 7 12 3	Case 1: 68 Case 2: 91

1071 - Baker Vai

All of you must have heard the name of Baker Vai. Yes, he rides a bike and likes to help people. That's why he is popular amongst general people.

Baker Vai lives in a city which can be modeled as a 2D $m \times n$ matrix. Where the north-west corner is cell $1, 1$ and the south-east corner is cell m, n . In each cell there are certain amount of people who needs help which is already known to Baker Vai.

Each day Baker Vai starts his journey from the north-west corner and he can only go to east or south. This way he reaches the south-east corner of the city. After that he returns back to the north-west, but this time he can only move to west or north. He doesn't want a cell to be visited twice other than the two corners. And if he visits a cell, he helps all the people in the cell.

Now you are given the map of the city and the number of people who need help in all cells for a particular day. You have to help Baker Vai finding the maximum number of people he can help in that day.

Input

Input starts with an integer T (≤ 25), denoting the number of test cases.

Each case contains a blank line and two integers, m, n ($2 \leq m, n \leq 100$). Each of the next m lines will contain n integers, denoting the number of people who are in need. In a cell there will be no more than 20 people and a cell can be empty, too.

Output

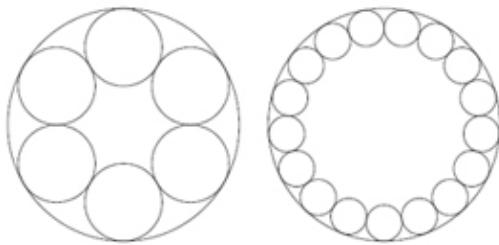
For each test case, print the case number and the maximum number of people Baker Vai can help considering the above conditions.

Sample Input	Output for Sample Input
2 3 3 1 1 1 1 0 1 1 1 1 3 4 1 1 0 1 1 1 1 1 0 1 10 1	Case 1: 8 Case 2: 18

1072 - Calm Down

George B. wants to be more than just a good American. He wants to make his daddy proud and become a hero. You know, like Shakib Khan.

But sneaky as he is, he wants a special revolver that will allow him to shoot more often than just the usual six times. This way he can fool and kill the enemy easily (at least that's what he thinks, and that's the best he can think). George has kidnapped . . . uh, I mean . . . "invited" you and will only let you go if you help him with the math. The piece of the revolver that contains the bullets looks like this (examples for 6 and 17 bullets):



There is a large circle with radius **R** and **n** little circles each having radius **r**, are placed inside on the border of the large circle. George wants his bullets to be as large as possible, so there should be no space between the circles. George will decide how large the whole revolver will be and how many bullets it shall contain. Your job is, given **R** and **n**, to compute **r**. You have decided to help, because you know that an idiot can't make a revolver even if you help him with the math.

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case contains a real number **R** ($0 < R < 1000$) and contains up to at most two places after the decimal point) and an integer **n** ($2 \leq n \leq 100$).

Output

For each test case, print the case number and **r** in a single line. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 4.0 6 4.0 17 3.14 100 42 2	Case 1: 1.3333333333 Case 2: 0.6209067545 Case 3: 0.0956260953 Case 4: 21

1073 - DNA Sequence

You are given a list of strings over the alphabet A (for adenine), C (cytosine), G (guanine), and T (thymine), and your task is to find the shortest string (which is typically not listed) that contains all given strings as substrings. If there are several such strings of shortest length, find the smallest in alphabetical/lexicographical order.

Input

Input starts with an integer **T** (≤ 35), denoting the number of test cases.

Each case starts with an integer denoting the number of strings **n** ($1 \leq n \leq 15$) in a single line. Then these **n** strings ($1 \leq \text{length} \leq 100$) follow, one on each line, and they consist of the letters '**A**', '**C**', '**G**' and '**T**' only.

Output

For each case, print the case number and the shortest (and lexicographically smallest) string according to the description above.

Sample Input	Output for Sample Input
2 2 TGCACA CAT 3 TAC ACT CTA	Case 1: TGCACAT Case 2: ACTAC

1074 - Extended Traffic

Dhaka city is getting crowded and noisy day by day. Certain roads always remain blocked in congestion. In order to convince people avoid shortest routes, and hence the crowded roads, to reach destination, the city authority has made a new plan. Each junction of the city is marked with a positive integer (≤ 20) denoting the busyness of the junction. Whenever someone goes from one junction (the source junction) to another (the destination junction), the city authority gets the amount (**busyness of destination - busyness of source**)³ (that means the cube of the difference) from the traveler. The authority has appointed you to find out the minimum total amount that can be earned when someone intelligent goes from a certain junction (the zero point) to several others.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

Each case contains a blank line and an integer **n** ($1 < n \leq 200$) denoting the number of junctions. The next line contains **n** integers denoting the busyness of the junctions from **1** to **n** respectively. The next line contains an integer **m**, the number of roads in the city. Each of the next **m** lines (one for each road) contains two junction-numbers (source, destination) that the corresponding road connects (all roads are unidirectional). The next line contains the integer **q**, the number of queries. The next **q** lines each contain a destination junction-number. There can be at most one direct road from a junction to another junction.

Output

For each case, print the case number in a single line. Then print **q** lines, one for each query, each containing the minimum total earning when one travels from junction **1** (the zero point) to the given junction. However, for the queries that gives total earning less than **3**, or if the destination is not reachable from the zero point, then print a '**?**'.

Sample Input	Output for Sample Input
<pre> 2 5 6 7 8 9 10 6 1 2 2 3 3 4 1 5 5 4 4 5 2 4 5 2 10 10 1 1 2 1 2 </pre>	<pre> Case 1: 3 4 Case 2: ?</pre>

1075 – Finding Routes

Shakil has been the greatest boy-scout in Bangladesh and has become quite a superstar because he always organized the most wonderful scavenger hunts (you know, where the kids have to find a certain route following certain hints). Shakil has retired now, but a nationwide election quickly found a successor for him, a guy called Nabil. He does a poor job, though, and wants to learn from Shakil's routes. Unfortunately Shakil has left only a few notes for his successor.

Shakil never wrote down his routes completely, he only left lots of little sheets on which he had written two consecutive steps of the routes. He then mixed these sheets and memorized his routes similarly to how some people learn for exams: practicing again and again, always reading the first step and trying to remember the following. This made much sense, since one step always required something from the previous step. Nabil however would like to have a route written down as one long sequence of all the steps in the correct order. Please help him make the nation happy again by reconstructing the routes. Shakil never visited a single place twice.

Input

Input starts with an integer **T (≤ 80)**, denoting the number of test cases.

Each case contains one route and its first line tells you how many steps (**$3 \leq S \leq 333$**) the route has. The next **S - 1** lines, each contains one consecutive pair of the steps on the route separated by a single space. The name of each step is always a single string of letters.

Output

For each case, print the case number in a single line. Then print **S** lines containing the steps of the route in correct order. Then print a blank line after each case.

Sample Input	Output for Sample Input
2 4 SwimmingPool OldTree BirdsNest Garage Garage SwimmingPool 3 Toilet Hospital VideoGame Toilet	Case 1: BirdsNest Garage SwimmingPool OldTree Case 2: VideoGame Toilet Hospital

1076 – Get the Containers

A conveyor belt has a number of vessels of different capacities each filled to brim with milk. The milk from conveyor belt is to be filled into '**m**' containers. The constraints are:

1. Whenever milk from a vessel is poured into a container, the milk in the vessel must be completely poured into that container only. That is milk from same vessel cannot be poured into different containers.
2. The milk from the vessel must be poured into the container in order which they appear in the conveyor belt. That is, you cannot randomly pick up a vessel from the conveyor belt and fill the container.
3. The **ith** container must be filled with milk only from those vessels that appear earlier to those that fill **jth** container, for all **i < j**.

Given the number of containers **m**, you have to fill the containers with milk from all the vessels, without leaving any milk in the vessel. The containers need not necessarily have same capacity. You are given the liberty to assign any possible capacities to them. Your job is to find out the minimal possible capacity of the container which has maximal capacity.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains two integers **n ($1 \leq n \leq 1000$)**, the number of vessels in the conveyor belt and then **m ($1 \leq m \leq 10^6$)**, which specifies the number of containers to which you have to transfer the milk. The next line contains the capacity **c ($1 \leq c \leq 10^6$)** of each vessel in order which they appear in the conveyor belt. Note that, milk is filled to the brim of any vessel. So the capacity of the vessel is equal to the amount of milk in it.

Output

For each case, print the case number and the desired result. See the samples for exact formatting.

Sample Input	Output for Sample Input
2 5 3 1 2 3 4 5 3 2 4 78 9	Case 1: 6 Case 2: 82

Note

For the first case, the capacities of the three containers be 6, 4 and 5. So, we can pour milk from the first three vessels to the first container and the rest in other two containers. So, the maximum capacity of the container is 6. Suppose the capacities of the containers be 3, 7 and 5. Then we can also pour the milk, however, the maximum capacity is 7. As we want to find the result, where the maximum capacity is as low as possible; the result is 6.

1077 – How Many Points?

Given two points **A** and **B** on the X-Y plane, output the number of the lattice points on the segment **AB**. Note that **A** and **B** are also lattice point. Those who are confused with the definition of lattice point, lattice points are those points which have both x and y co-ordinate as integer.

For example, for **A** (3, 3) and **B** (-1, -1) the output is 5. The points are: (-1, -1), (0, 0), (1, 1), (2, 2) and (3, 3).

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case contains four integers, **A_x**, **A_y**, **B_x** and **B_y**. Each of them will be fit into a 32 bit signed integer.

Output

For each test case, print the case number and the number of lattice points between **AB**.

Sample Input	Output for Sample Input
2 3 3 -1 -1 0 0 5 2	Case 1: 5 Case 2: 2

1078 – Integer Divisibility

If an integer is not divisible by 2 or 5, some multiple of that number in decimal notation is a sequence of only a digit. Now you are given the number and the only allowable digit, you should report the number of digits of such multiple.

For example you have to find a multiple of 3 which contains only 1's. Then the result is 3 because is 111 (3-digit) divisible by 3. Similarly if you are finding some multiple of 7 which contains only 3's then, the result is 6, because 333333 is divisible by 7.

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case will contain two integers **n** ($0 < n \leq 10^6$ and **n** will not be divisible by 2 or 5) and the allowable digit ($1 \leq \text{digit} \leq 9$).

Output

For each case, print the case number and the number of digits of such multiple. If several solutions are there; report the minimum one.

Sample Input	Output for Sample Input
3 3 1 7 3 9901 1	Case 1: 3 Case 2: 6 Case 3: 12

1079 - Just another Robbery

As Harry Potter series is over, Harry has no job. Since he wants to make quick money, (he wants everything quick!) so he decided to rob banks. He wants to make a calculated risk, and grab as much money as possible. But his friends - Hermione and Ron have decided upon a tolerable probability **P** of getting caught. They feel that he is safe enough if the banks he robs together give a probability less than **P**.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains a real number **P**, the probability Harry needs to be below, and an integer **N** ($0 < N \leq 100$), the number of banks he has plans for. Then follow **N** lines, where line **j** gives an integer **M_j** ($0 < M_j \leq 100$) and a real number **P_j**. Bank **j** contains **M_j** millions, and the probability of getting caught from robbing it is **P_j**. A bank goes bankrupt if it is robbed, and you may assume that all probabilities are independent as the police have very low funds.

Output

For each case, print the case number and the maximum number of millions he can expect to get while the probability of getting caught is less than **P**.

Sample Input	Output for Sample Input
3 0.04 3 1 0.02 2 0.03 3 0.05 0.06 3 2 0.03 2 0.03 3 0.05 0.10 3 1 0.03 2 0.02 3 0.05	Case 1: 2 Case 2: 4 Case 3: 6

Note

For the first case, if he wants to rob bank 1 and 2, then the probability of getting caught is **0.02 + (1 - 0.02) * .03 = 0.0494** which is greater than the given probability (**0.04**). That's why he has only option, just to rob rank 2.

1080 - Binary Simulation

Given a binary number, we are about to do some operations on the number. Two types of operations can be here.

'I i j' which means invert the bit from i to j (inclusive)

'Q i' answer whether the i^{th} bit is 0 or 1

The MSB (most significant bit) is the first bit (**i.e. $i=1$**). The binary number can contain leading zeroes.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a line containing a binary integer having length **n ($1 \leq n \leq 10^5$)**. The next line will contain an integer **q ($1 \leq q \leq 50000$)** denoting the number of queries. Each query will be either in the form '**I i j**' where i, j are integers and $1 \leq i \leq j \leq n$. Or the query will be in the form '**Q i**' where i is an integer and $1 \leq i \leq n$.

Output

For each case, print the case number in a single line. Then for each query '**Q i**' you have to print 1 or 0 depending on the i^{th} bit.

Sample Input	Output for Sample Input
2 0011001100 6 I 1 10 I 2 7 Q 2 Q 1 Q 7 Q 5 1011110111 6 I 1 10 I 2 7 Q 2 Q 1 Q 7 Q 5	Case 1: 0 1 1 0 Case 2: 0 0 0 1

Note

Dataset is huge, use faster i/o methods.

1081 – Square Queries

Little Tommy is playing a game. The game is played on a 2D $N \times N$ grid. There is an integer in each cell of the grid. The rows and columns are numbered from 1 to N .

At first the board is shown. When the user presses a key, the screen shows three integers I, J, S which designates a square (I, J) to $(I+S-1, J+S-1)$ in the grid. The player has to predict the largest integer found in this square. The user will be given points based on the difference between the actual result and the given result.

Tommy doesn't like to lose. So, he made a plan, he will take help of a computer to generate the result. But since he is not a good programmer, he is seeking your help.

Input

Input starts with an integer T (≤ 3), denoting the number of test cases.

The first line of a case is a blank line. The next line contains two integers N ($1 \leq N \leq 500$), Q ($0 \leq Q \leq 50000$). Each of the next N lines will contain N space separated integers forming the grid. All the integers will be between 0 and 10^5 .

Each of the next Q lines will contain a query which is in the form $I\ J\ S$ ($1 \leq I, J \leq N$ and $1 \leq I + S, J + S < N$ and $S > 0$).

Output

For each test case, print the case number in a single line. Then for each query you have to print the maximum integer found in the square whose top left corner is (I, J) and whose bottom right corner is $(I+S-1, J+S-1)$.

Sample Input	Output for Sample Input
1 4 5 67 1 2 3 8 88 21 1 89 12 0 12 5 5 5 5 1 1 2 1 3 2 3 3 2 1 1 4 2 2 3	Case 1: 88 21 12 89 88

Note

Dataset is huge. Use faster i/o methods.

1082 – Array Queries

Given an array with N elements, indexed from 1 to N . Now you will be given some queries in the form $I J$, your task is to find the minimum value from index I to J .

Input

Input starts with an integer T (≤ 5), denoting the number of test cases.

The first line of a case is a blank line. The next line contains two integers N ($1 \leq N \leq 10^5$), q ($1 \leq q \leq 50000$). The next line contains N space separated integers forming the array. These integers range in $[0, 10^5]$.

The next q lines will contain a query which is in the form $I J$ ($1 \leq I \leq J \leq N$).

Output

For each test case, print the case number in a single line. Then for each query you have to print a line containing the minimum value between index I and J .

Sample Input	Output for Sample Input
2 5 3 78 1 22 12 3 1 2 3 5 4 4 1 1 10 1 1	Case 1: 1 3 12 Case 2: 10

Note

Dataset is huge. Use faster I/O methods.

1083 – Histogram

A histogram is a polygon composed of a sequence of rectangles aligned at a common base line. The rectangles have equal widths but may have different heights. For example, the figure shows the histogram that consists of rectangles with the heights 2, 1, 4, 5, 1, 3, 3 measured in units where the width of the rectangles is 1.



Usually, histograms are used to represent discrete distributions, e.g., the frequencies of characters in texts. Note that the order of the rectangles, i.e., their heights, is important. Calculate the area of the largest rectangle in a histogram that is aligned at the common base line, too. The figure on the right shows the largest aligned rectangle for the depicted histogram.

Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

Each case contains a line with an integer **N** ($1 \leq N \leq 30000$) denoting the number of rectangles. The next line contains **N** space separated positive integers (≤ 30000) denoting the heights.

Output

For each case, print the case number and the largest rectangle that can be made.

Sample Input	Output for Sample Input
2 7 2 1 4 5 1 3 3 5 4 4 3 2 4	Case 1: 8 Case 2: 10

Note

Dataset is huge; use faster I/O methods.

1084 – Winter

Winter is coming. In a land far away, N men are spending the nights in a valley in a largest field. The valley is so narrow that it can be considered to be a straight line running east-to-west.

Although standing in the valley does shield them from the wind, the group still shivers during the cold nights. They, like anyone else, would like to gather together for warmth.

Near the end of each day, each man i finds himself somewhere in the valley at a unique location L_i . The men want to gather into groups of three or more persons since two persons just aren't warm enough. They want to be in groups before sunset, so the distance K each man can walk to form a group is limited. Determine the smallest number of groups the men can form.

Input

Input starts with an integer T (≤ 15), denoting the number of test cases.

Each case starts with two integers N ($1 \leq N \leq 10^5$) and K ($1 \leq K \leq 10^6$). Each of the next N line contains an integer L_i ($1 \leq L_i \leq 10^8$).

Output

For each case, print the case number and smallest number of groups the men can gather into. If there is no way for all the men to gather into groups of at least size three, output **-1**.

Sample Input	Output for Sample Input
2 6 10 2 10 15 13 28 9 3 1 1 10 20	Case 1: 2 Case 2: -1

Note

Dataset is huge, use faster I/O methods.

1085 – All Possible Increasing Subsequences

An increasing subsequence from a sequence $A_1, A_2 \dots A_n$ is defined by $A_{i1}, A_{i2} \dots A_{ik}$, where the following properties hold

1. $i_1 < i_2 < i_3 < \dots < i_k$ and
2. $A_{i1} < A_{i2} < A_{i3} < \dots < A_{ik}$

Now you are given a sequence, you have to find the number of all possible increasing subsequences.

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case contains an integer n ($1 \leq n \leq 10^5$) denoting the number of elements in the initial sequence. The next line will contain n integers separated by spaces, denoting the elements of the sequence. Each of these integers will be fit into a 32 bit signed integer.

Output

For each case of input, print the case number and the number of possible increasing subsequences modulo **1000000007**.

Sample Input	Output for Sample Input
3 3 1 1 2 5 1 2 1000 1000 1001 3 1 10 11	Case 1: 5 Case 2: 23 Case 3: 7

Notes

1. For the first case, the increasing subsequences are (1), (1, 2), (1), (1, 2), 2.
2. Dataset is huge, use faster I/O methods.

1086 – Jogging Trails

Robin is training for a marathon. Behind his house is a park with a large network of jogging trails connecting water stations. Robin wants to find the shortest jogging route that travels along every trail at least once.

For each case, there should be one line of output giving the length of Robin's jogging route.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains two positive integers **n ($2 \leq n \leq 15$)**, the number of water stations, and **m ($0 \leq 1000$)**, the number of trails. For each trail, there is one subsequent line of input containing three positive integers: the first two, between **1** and **n**, indicating the water stations at the end points of the trail; the third indicates the length of the trail, in cubits. There may be more than one trail between any two stations; each different trail is given only once in the input; each trail can be travelled in either direction. It is possible to reach any trail from any other trail by visiting a sequence of water stations connected by trails. Robin's route may start at any water station, and must end at the same station.

Output

For each case, print the case number and the minimum possible length of Robin's jogging route

Sample Input	Output for Sample Input
1 4 5 1 2 3 2 3 4 3 4 5 1 4 10 1 3 12	Case 1: 41

1087 - Diablo

All of you must have played the game 'Diablo'. It's an exclusive game to play. In this game the main opponent of you is Diablo. If you kill him the game finishes. But as usual, Diablo is smarter than you.

Diablo has a large number of army. Diablo arranges them in a line in any arbitrary order and everyone is given an integer id. Each time Diablo either adds one army in the end or he calls for the k^{th} army (from left) from the line. Then the army gets out and it attacks you.

Since you are a great magician, you can read Diablo's mind. Now you want to find the id of the armies who are about to attack you.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

The first line of each case is a blank line. The next line contains two integers **n** ($0 \leq n \leq 10^5$), denoting the number of the initial army and **q** ($1 \leq q \leq 50000$) representing the number of queries. The next line contains **n** space separated integers. The i^{th} integer of this line denotes the id of the i^{th} person. Each of these integers will be positive and fits into a 32 bit signed integer. Each of the next **q** lines will contain a query, of the form:

a p (add a person at the end of the line whose id is **p**)

c k (call the k^{th} person from the line (from left), **k** is a positive 32 bit signed integer)

Output

For each case of input, print the case number in a line. Then for all the queries '**c k**' you have to print the **id** of the k^{th} person or '**none**' if there is none.

Sample Input	Output for Sample Input
2 5 5 6 5 3 2 1 c 1 c 1 a 20 c 4 c 4 2 1 18811 1991 c 1	Case 1: 6 5 20 none Case 2: 18811

Notes

Dataset is huge, use faster i/o methods.

1088 – Points in Segments

Given n points (1 dimensional) and q segments, you have to find the number of points that lie in each of the segments. A point p_i will lie in a segment $A \leq p_i \leq B$.

For example if the points are **1, 4, 6, 8, 10**. And the segment is **0** to **5**. Then there are **2** points that lie in the segment.

Input

Input starts with an integer T (≤ 5), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 10^5$) and q ($1 \leq q \leq 50000$). The next line contains n space separated integers denoting the points in ascending order. All the integers are distinct and each of them range in $[0, 10^8]$.

Each of the next q lines contains two integers $A_k \ B_k$ ($0 \leq A_k \leq B_k \leq 10^8$) denoting a segment.

Output

For each case, print the case number in a single line. Then for each segment, print the number of points that lie in that segment.

Sample Input	Output for Sample Input
1 5 3 1 4 6 8 10 0 5 6 10 7 100000	Case 1: 2 3 2

Note

Dataset is huge, use faster I/O methods.

1089 - Points in Segments (II)

Given n segments (1 dimensional) and q points, for each point you have to find the number of segments which contain that point. A point p_i will lie in a segment $A \ B$ if $A \leq p_i \leq B$.

For example, if the segments are $(6 \ 12)$, $(8 \ 8)$, $(10 \ 12)$, $(8 \ 11)$, $(0 \ 12)$ and the point is 11 , then it is contained by 4 segments.

Input

Input starts with an integer T (≤ 5), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 50000$) and q ($1 \leq q \leq 50000$).

Each of the next n lines contains two integers $A_k \ B_k$ ($0 \leq A_k \leq B_k \leq 10^8$) denoting a segment.

Each of the next q lines contains an integer denoting a point. Each of them range in $[0, 10^8]$.

Output

For each case, print the case number in a single line. Then for each point, print the number of segments that contain that point.

Sample Input	Output for Sample Input
1 5 4 6 12 8 8 10 12 8 11 0 12 11 12 2 20	Case 1: 4 3 1 0

Notes

Dataset is huge, use faster I/O methods.

1090 – Trailing Zeroes (II)

Find the number of trailing zeroes for the following function:

$${}^nC_r * p^q$$

where n, r, p, q are given. For example, if $n = 10, r = 4, p = 1, q = 1$, then the number is 210 so, number of trailing zeroes is 1.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case contains four integers: n, r, p, q ($1 \leq n, r, p, q \leq 10^6, r \leq n$).

Output

For each test case, print the case number and the number of trailing zeroes.

Sample Input	Output for Sample Input
2 10 4 1 1 100 5 40 5	Case 1: 1 Case 2: 6

1091 – The Fastest Sorting Ever!

Sorting algorithms are essentially necessary in computer science. If there are **n** numbers then the best time to sort the numbers is **O(n log n)**. Here you are asked to write a code that will sort the numbers in ascending order using only 'if' and 'else' statements.

In your code, you should attach the function to print (as in sample) the numbers. In your codes you can use spaces freely, but be careful to use spaces in the printing format (as in the sample **printf**). It should be exact. And no lines should be empty.

Constraints

1. Your code should contain a function named **print()** which prints the variables. You can use spaces freely but the format in **printf** should be correct. There should be **n** parameters for **print()** and they are **a, b, c, ...** (ascending order). The **printf** (as in **C**) should print all of them in ascending order. First three lines should be the print function. See the samples.
2. Your code should contain a function named **sort()**. The parameters should be same as **print()**. It should contain three kinds of statements
 - a. **if(x < y)**
 - b. **else**
 - c. **print(a, b, ...);**
3. Every statement should be in a single line. And no line should be empty.
4. Nested **if** statements are allowed. And every **if** statement should have a corresponding **else** statement.
5. Assume that the inputs given to your code will be distinct. That means for any two given inputs **x, y** you can assume that **x ≠ y**.
6. For a code for **n** variables there should be exactly **n! * 3 + 3** lines (excluding the case).
7. Check the samples very carefully for other clarifications.

Input

Input starts with an integer **T (≤ 7)**, denoting the number of test cases.

Each case contains an integer **n (1 ≤ n ≤ 7)** denoting the numbers that will be given as input to your code.

Output

For each case, print the case number in a line. Then you have to print your code. Check the samples for the formatting details. There can be multiple solutions, print any valid solution.

Sample Input	Output for Sample Input
<pre>3 1 2 3</pre>	<pre>Case 1: void print(int a) { printf("%d\n", a); } void sort(int a) { print(a); } Case 2: void print(int a,int b){ printf("%d %d\n",a,b); } void sort(int a, int b){ if(b < a) print(b, a); else print(a, b); } Case 3: void print(int a, int b, int c) { printf("%d %d %d\n", a, b, c); } void sort(int a, int b, int c) { if(a < b) if(a < c) if(b < c) print(a, b, c); else print(a, c, b); else print(c, a, b); else if(b < c) if(a < c) print(b, a, c); else print(b, c, a); else print(c, b, a); }</pre>

1092 - Lighted Panels

You are given an $\mathbf{R} \times \mathbf{C}$ 2D grid consisting of several light panels. Each cell contains either a '*' or a '!'. '*' means the panel is on, and '!' means it's off. If you touch a panel, its state will be toggled. That means, if you touch a panel that's on, it will turn off, and if you touch a panel that's off, it will turn on. But if we touch a panel, all its horizontal, vertical, and diagonal **adjacent** panels will also toggle their states.

Now you are given the configuration of the grid. Your goal is to turn on all the lights. Print the minimum number of touches required to achieve this goal.

Input

Input starts with an integer \mathbf{T} (≤ 125), denoting the number of test cases.

Each test case starts with two integers \mathbf{R} ($1 \leq \mathbf{R} \leq 8$) and \mathbf{C} ($1 \leq \mathbf{C} \leq 8$). Then there will be \mathbf{R} lines each containing \mathbf{C} characters ('*' or '!').

Output

For each test case, print the case number and the minimum number of touches required to have all the light panels in the board on at the same time. If it is not possible then print "**impossible**".

Sample Input	Output for Sample Input
4 5 5 ***** *...* *...* *...* ***** 1 2 . * 3 3 **. **. . . . 4 4 *... **.. . ** . . *	Case 1: 1 Case 2: impossible Case 3: 2 Case 4: 10

1093 – Ghajini

Amir is having a short term memory problem. He can't remember anything for more than **d** milliseconds.

Amir is playing a game named 'Find Max Difference'. The game is actually designed for children. There is a screen which shows an integer for 1 millisecond. In the very next millisecond the screen shows another integer. The target of the game is to find the maximum difference of any two numbers shown in the screen.

But soon Amir found that the game is more difficult for him, because his short term memory problem. So, he uses a paper to write the maximum difference he has found so far. So, Amir wants your help. You have to write a program to help Amir.

Input

Input starts with an integer **T (≤ 5)**, denoting the number of test cases.

Each case starts with two integers **n ($2 \leq n \leq 10^5$)**, **d ($1 \leq d \leq n$)**, **n** means the total number of integers the screen will show. The next line contains **n** space separated integers in range **[0, 10^8]**.

Output

For each case, print the case number and the maximum difference found by Amir.

Sample Input	Output for Sample Input
3 6 2 6 0 8 8 8 4 8 3 19 8 4 13 12 1 0 13 2 2 1 1	Case 1: 8 Case 2: 15 Case 3: 0

Notes

Dataset is huge, use faster I/O methods.

1094 - Farthest Nodes in a Tree

Given a tree (a connected graph with no cycles), you have to find the farthest nodes in the tree. The edges of the tree are weighted and undirected. That means you have to find two nodes in the tree whose distance is maximum amongst all nodes.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with an integer **n** ($2 \leq n \leq 30000$) denoting the total number of nodes in the tree. The nodes are numbered from **0** to **n-1**. Each of the next **n-1** lines will contain three integers **u v w** ($0 \leq u, v < n, u \neq v, 1 \leq w \leq 10000$) denoting that node **u** and **v** are connected by an edge whose weight is **w**. You can assume that the input will form a valid tree.

Output

For each case, print the case number and the maximum distance.

Sample Input	Output for Sample Input
2 4 0 1 20 1 2 30 2 3 50 5 0 2 20 2 1 10 0 3 29 0 4 50	Case 1: 100 Case 2: 80

Notes

Dataset is huge, use faster i/o methods.

1095 - Arrange the Numbers

Consider this sequence $\{1, 2, 3 \dots N\}$, as an initial sequence of first N natural numbers. You can rearrange this sequence in many ways. There will be a total of $N!$ arrangements. You have to calculate the number of arrangement of first N natural numbers, where in first M positions; exactly K numbers are in their initial position.

For Example, $N = 5, M = 3, K = 2$

You should count this arrangement $\{1, 4, 3, 2, 5\}$, here in first 3 positions 1 is in 1st position and 3 in 3rd position. So exactly 2 of its first 3 are in there initial position.

But you should not count $\{1, 2, 3, 4, 5\}$.

Input

Input starts with an integer T (≤ 1000), denoting the number of test cases.

Each case contains three integers N ($1 \leq N \leq 1000$), M ($M \leq N$), K ($0 < K \leq M$).

Output

For each case, print the case number and the total number of possible arrangements modulo 1000000007 .

Sample Input	Output for Sample Input
2 5 3 2 10 6 3	Case 1: 12 Case 2: 64320

1096 - nth Term

You have to find the n^{th} term of the following function:

$$\begin{aligned}f(n) &= a * f(n-1) + b * f(n-3) + c, \text{ if } (n > 2) \\&= 0, \text{ if } (n \leq 2)\end{aligned}$$

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case contains four integers n ($0 \leq n \leq 10^8$), a b c ($1 \leq a, b, c \leq 10000$).

Output

For each case, print the case number and $f(n)$ modulo 10007 .

Sample Input	Output for Sample Input
2 10 1 2 3 5 1 3 9	Case 1: 162 Case 2: 27

1097 - Lucky Number

Lucky numbers are defined by a variation of the well-known sieve of Eratosthenes. Beginning with the natural numbers strike out all even ones, leaving the odd numbers **1, 3, 5, 7, 9, 11, 13, ...** The second number is **3**, next strike out every third number, leaving **1, 3, 7, 9, 13, ...** The third number is **7**, next strike out every seventh number and continue this process infinite number of times. The numbers surviving are called lucky numbers. The first few lucky numbers are:

1, 3, 7, 9, 13, 15, 21, 25, 31, 33, ...

In this problem your task is to find the n^{th} lucky number where n is given in input.

Input

Input starts with an integer **T (≤ 10000)**, denoting the number of test cases.

Each case contains an integer **n ($1 \leq n \leq 10^5$)**.

Output

For each case, print the case number and the n^{th} lucky number.

Sample Input	Output for Sample Input
2 2 100000	Case 1: 3 Case 2: 1429431

1098 – A New Function

We all know that any integer number **n** is divisible by **1** and **n**. That is why these two numbers are not the actual divisors of any numbers. The function **SOD(n)** (sum of divisors) is defined as the summation of all the actual divisors of an integer number **n**. For example,

$$\text{SOD}(24) = 2+3+4+6+8+12 = 35.$$

The function **CSOD(n)** (cumulative **SOD**) of an integer **n**, is defined as below:

$$CSOD(n) = \sum_{i=1}^n SOD(i)$$

Given the value of **n**, your job is to find the value of **CSOD(n)**.

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case contains an integer **n** ($0 \leq n \leq 2 * 10^9$).

Output

For each case, print the case number and the result. You may assume that each output will fit into a **64** bit signed integer.

Sample Input	Output for Sample Input
3 2 100 200000000	Case 1: 0 Case 2: 3150 Case 3: 12898681201837053

1099 – Not the Best

Robin has moved to a small village and sometimes enjoys returning to visit one of his best friends. He does not want to get to his old home too quickly, because he likes the scenery along the way. He has decided to take the second-shortest rather than the shortest path. He knows there must be some second-shortest path.

The countryside consists of **R** bidirectional roads, each linking two of the **N** intersections, conveniently numbered from **1** to **N**. Robin starts at intersection **1**, and his friend (the destination) is at intersection **N**.

The second-shortest path may share roads with any of the shortest paths, and it may backtrack i.e., use the same road or intersection more than once. The second-shortest path is the shortest path whose length is longer than the shortest path(s) (i.e., if two or more shortest paths exist, the second-shortest path is the one whose length is longer than those but no longer than any other path).

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case contains two integers **N** ($1 \leq N \leq 5000$) and **R** ($1 \leq R \leq 10^5$). Each of the next **R** lines contains three space-separated integers: **u**, **v** and **w** that describe a road that connects intersections **u** and **v** and has length **w** ($1 \leq w \leq 5000$).

Output

For each case, print the case number and the second best shortest path as described above.

Sample Input	Output for Sample Input
2 3 3 1 2 100 2 3 200 1 3 50 4 4 1 2 100 2 4 200 2 3 250 3 4 100	Case 1: 150 Case 2: 450

1100 – Again Array Queries

Given an array with n integers, and you are given two indices i and j ($i \neq j$) in the array. You have to find two integers in the range whose difference is minimum. You have to print this value. The array is indexed from 0 to $n-1$.

Input

Input starts with an integer T (≤ 5), denoting the number of test cases.

Each case contains two integers n ($2 \leq n \leq 10^5$) and q ($1 \leq q \leq 10000$). The next line contains n space separated integers which form the array. These integers range in $[1, 1000]$.

Each of the next q lines contains two integers i and j ($0 \leq i < j < n$).

Output

For each test case, print the case number in a line. Then for each query, print the desired result.

Sample Input	Output for Sample Input
2 5 3 10 2 3 12 7 0 2 0 4 2 4 2 1 1 2 0 1	Case 1: 1 1 4 Case 2: 1

Notes

Dataset is huge, use faster I/O methods.

1101 - A Secret Mission

All of you have heard about Evil Jack Diablo, the one who had stolen the whole problem set from the Good Judges last time. Once again he is making evil plans, but he does not know that Alice is on a secret mission. There will be several pairs of City of Evils on the way of Alice's mission.

There will be **N** evil cities (numbered by **1, 2, ..., N**) connected by **M** bidirectional roads. There will be evil guards patrolling the roads. Since they are not much intelligent, the danger of travelling in each road is not the same. Alice is going to travel from city **s** to city **t**. You can safely assume that it's possible to travel from any city to another. John the legend programmer has estimated the danger of each road but since he is busy arranging contests, Alice is depending on you now. Danger of a path from city **s** to city **t** is defined as the maximum danger of any road on this path. As you are one of the most talented programmers, you will love to help Alice by finding the least dangerous paths to prevent Evil Jack from doing harms to the Judges.

Input

Input starts with an integer **T (≤ 3)**, denoting the number of test cases.

Each case contains two integers **N, M ($2 \leq N \leq 50000, 1 \leq M \leq 10^5$)** denoting the number of cities and roads. Each of the next **M** lines contains three integers: **x_i, y_i, d_i ($1 \leq x_i, y_i \leq N, x_i \neq y_i, 1 \leq d_i \leq 10^4$)** - the cities connected by the **ith** road and its degree of danger. Next line contains an integer **q ($1 \leq q \leq 50000$)**. Each of the next **q** lines contains two integers: **s_i and t_i ($1 \leq s_i, t_i \leq N, s_i \neq t_i$)**.

Output

For each test case, print the case number first. Then for each query **s_i t_i**, print the least dangerous path in a line.

Sample Input	Output for Sample Input
2 4 5 1 2 10 1 3 20 1 4 100 2 4 30 3 4 10 2 1 4 4 1 2 1 1 2 100 1 1 2	Case 1: 20 20 Case 2: 100

Notes

Dataset is huge, use faster i/o methods.

1102 - Problem Makes Problem

As I am fond of making easier problems, I discovered a problem. Actually, the problem is 'how can you make **n** by adding **k** non-negative integers?' I think a small example will make things clear. Suppose **n=4** and **k=3**. There are **15** solutions. They are

1. 0 0 4
2. 0 1 3
3. 0 2 2
4. 0 3 1
5. 0 4 0
6. 1 0 3
7. 1 1 2
8. 1 2 1
9. 1 3 0
10. 2 0 2
11. 2 1 1
12. 2 2 0
13. 3 0 1
14. 3 1 0
15. 4 0 0

As I have already told you that I use to make problems easier, so, you don't have to find the actual result. You should report the result modulo **1000,000,007**.

Input

Input starts with an integer **T** (≤ 25000), denoting the number of test cases.

Each case contains two integer **n** ($0 \leq n \leq 10^6$) and **k** ($1 \leq k \leq 10^6$).

Output

For each case, print the case number and the result modulo **1000000007**.

Sample Input	Output for Sample Input
4 4 3 3 5 1000 3 1000 5	Case 1: 15 Case 2: 35 Case 3: 501501 Case 4: 84793457

1103 – Castle Walls

In medieval times, knights commanded big armies of peasants. When they had to storm a castle they would line up neatly in front of the castle's wall and throw their grappling hooks over the walls. If one does not throw straight it can easily happen that two hooks cross, making it impossible for the two peasants to climb the wall. That's why every knight made his peasants practice a lot so that this would not happen in combat.

Due to the recent Sir Arthur-Madam Claire Marriage (ACM), two peasant armies have to be merged. Traditionally Sir Arthur's peasants wear blue and Madame Claire's peasants wear red. When practicing together, both armies mix up in front of a castle's wall. On Sir Arthur's command, they all throw their grappling hooks. Due to their perfect training the hooks will never cross within an army, however it can happen that a hook thrown by a blue peasant crosses one thrown by a peasant of the red army.

For statistical purposes, Sir Arthur now needs to find out how many grappling hooks have crossed so that he can measure how well their armies have already been merged. Given the positions of blue and red peasants as well as the positions they threw their grappling hooks at, determine how many distinct pairs of blue and red peasants crossed their hooks.

If there are n blue and m red peasants, the positions in the line where the peasants are standing are numbered from 1 to $n + m$. The positions on the castle's wall are numbered from 1 to $n + m$ as well, where position i is directly opposite of position i on the line the peasants are standing on. A grappling hook thrown from position i to j is said to cross another hook thrown from p to q if and only if

$$(i < p \text{ and } j \geq q) \text{ or } (i > p \text{ and } j \leq q)$$

Grappling hooks of the same color will never cross each other, nor will two peasants occupy the same position on the line. However, two hooks (of different color) can be thrown to the same position in which case they are said to cross each other as well.

Input

Input starts with an integer T (≤ 8), denoting the number of test cases.

Each case starts with a line containing two integers n and m , the number of blue and red peasants, respectively ($1 \leq n, m \leq 10^5$). The next n lines describe the blue peasants followed by m more lines for the red peasants. Each line consists of two integer i and j separated by a space, indicating the peasant's position i and the position j he threw his grappling hook to ($1 \leq i, j \leq n + m$).

Output

For each test case, print the case number and the number of distinct pairs of peasants whose grappling hooks are crossed.

Sample Input	Output for Sample Input
2 2 2 1 2 3 4 2 1 4 3 2 3 1 3 2 5 5 3 3 1 4 2	Case 1: 2 Case 2: 6

1104 - Birthday Paradox

Sometimes some mathematical results are hard to believe. One of the common problems is the birthday paradox. Suppose you are in a party where there are **23** people including you. What is the probability that at least two people in the party have same birthday? Surprisingly the result is more than **0.5**. Now here you have to do the opposite. You have given the number of days in a year. Remember that you can be in a different planet, for example, in Mars, a year is **669** days long. You have to find the minimum number of people you have to invite in a party such that the probability of at least two people in the party have same birthday is at least **0.5**.

Input

Input starts with an integer **T (≤ 20000)**, denoting the number of test cases.

Each case contains an integer **n ($1 \leq n \leq 10^5$)** in a single line, denoting the number of days in a year in the planet.

Output

For each case, print the case number and the desired result.

Sample Input	Output for Sample Input
2 365 669	Case 1: 22 Case 2: 30

1105 - Fi Binary Number

A Fi-binary number is a number that contains only 0 and 1. It does not contain any leading 0. And also it does not contain 2 consecutive 1. The first few such numbers are 1, 10, 100, 101, 1000, 1001, 1010, 10000, 10001, 10010, 10100, 10101 and so on. You are given n . You have to calculate the n^{th} Fi-Binary number.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case contains an integer n ($1 \leq n \leq 10^9$).

Output

For each case, print the case number and the n^{th} Fi-Binary number

Sample Input	Output for Sample Input
4	Case 1: 10010
10	Case 2: 101010
20	Case 3: 1010001
30	Case 4: 10001001
40	

1106 - Gone Fishing

John is going on a fishing trip. He has **h** hours available, and there are **n** lakes in the area all reachable along a single, one-way road. John starts at lake **1**, but he can finish at any lake he wants. He can only travel from one lake to the next one, but he does not have to stop at any lake unless he wishes to. For each **i** (**1 to n-1**), the number of 5-minute intervals it takes to travel from lake **i** to lake **i+1** is denoted **t_i**. For example, **t₃=4** means that it takes **20** minutes to travel from lake **3** to **4**.

To help plan his fishing trip, John has gathered some information about the lakes. For each lake **i**, the number of fish expected to be caught in the initial **5** minutes, denoted **f_i**, is known. Each **5** minutes of fishing decreases the number of fish expected to be caught in the next **5**-minute interval by a constant rate of **d_i**. If the number of fish expected to be caught in an interval is less than or equal to **d_i**, there will be no more fish left in the lake in the next interval. To simplify the planning, John assumes that no one else will be fishing at the lakes to affect the number of fish he expects to catch. Write a program to help John plan his fishing trip to maximize the number of fish expected to be caught. The number of minutes spent at each lake must be a multiple of **5**.

Input

Input starts with an integer **T** (**≤ 100**), denoting the number of test cases.

Each case starts with a line containing two integers **n** (**2 ≤ n ≤ 25**) and **h** (**1 ≤ h ≤ 16**). Next, there is a line of **n** integers specifying **f_i** (**0 ≤ f_i ≤ 1000**), then a line of **n** integers **d_i** (**0 ≤ d_i ≤ 1000**), and finally, a line of **n-1** integers denoting **t_i** (**0 < t_i < 192**).

Output

For each test case, print the case number first. Then print the number of minutes spent at each lake, separated by commas, for the plan achieving the maximum number of fish expected to be caught. This is followed by a line containing the number of fish expected. If multiple plans exist, choose the one that spends as long as possible at lake **1**. If there is still a tie, choose the one that spends as long as possible at lake **2**, and so on.

Sample Input	Output for Sample Input
3 2 1 10 1 2 5 2 4 4 10 15 20 17 0 3 4 3 1 2 3 4 4 10 15 50 30 0 3 4 3 1 2 3	Case 1: 45, 5 Number of fish expected: 31 Case 2: 240, 0, 0, 0 Number of fish expected: 480 Case 3: 115, 10, 50, 35 Number of fish expected: 724

1107 – How Cow

Mr Kopa Samsu is a farmer. He has a land of rectangular shape. But there are cows that disturb him a lot. The cows use to enter his land and ruin his crops. Now Mr Kopa Samsu has become smarter. He has a GPS system that will help him to know the position of the cows. So, you can think his land as a 2D grid, and cows can be treated as points. Now you are given the information of his land and cows. You have to tell him whether a cow is inside his land or not.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

The first line of each case contains four integers $x_1 \ y_1 \ x_2 \ y_2$, where (x_1, y_1) is the lower left coordinate of his land and (x_2, y_2) is the upper right coordinate of his land. You can assume that the sides of the land are axis parallel. The next line contains an integer **M** ($1 \leq M \leq 100$) denoting the number of cows. Each of the next **M** lines contains two integers each denoting $x \ y$ - the position of a cow. You can safely assume that no cow will lie on the boundary of the rectangle. All the coordinates will lie in the range $[0, 10000]$.

Output

For each case you have to print the case number in a line first. Then for each cow, you have to print 'Yes' or 'No' depending whether the cow is inside the land or not.

Sample Input	Output for Sample Input
1 1 2 8 10 7 0 0 5 6 1 0 7 9 3 5 10 10 1 11	Case 1: No Yes No Yes Yes No No

1108 - Instant View of Big Bang

Have you forgotten about wormholes? Oh my god! Ok, let me explain again.

A wormhole is a subspace tunnel through space and time connecting two star systems. Wormholes have a few peculiar properties:

1. Wormholes are one-way only.
2. The time it takes to travel through a wormhole is negligible.
3. A wormhole has two end points, each situated in a star system.
4. A star system may have more than one wormhole end point within its boundaries.
5. Between any pair of star systems, there is at most one wormhole in each direction.
6. There are no wormholes with both end points in the same star system.

All wormholes have a constant time difference between their end points. For example, a specific wormhole may cause the person traveling through it to end up 15 years in the future. Another wormhole may cause the person to end up 42 years in the past.

A brilliant physicist wants to use wormholes to study the Big Bang. Since warp drive has not been invented yet, it is not possible for her to travel from one star system to another one directly. This can be done using wormholes, of course.

The scientist can start her journey from any star system. Then she wants to reach a cycle of wormholes somewhere in the universe that causes her to end up in the past. By traveling along this cycle a lot of times, the scientist is able to go back as far in time as necessary to reach the beginning of the universe and see the Big Bang with her own eyes. Write a program to help her to find such star systems where she can start her journey.

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case starts with a blank line. The next line contains two numbers **n** and **m**. These indicate the number of star systems ($1 \leq n \leq 1000$) and the number of wormholes ($0 \leq m \leq 2000$). The star systems are numbered from **0** to **n-1**. For each wormhole a line containing three integer numbers **x**, **y** and **t** is given. These numbers indicate that this wormhole allows someone to travel from the star system numbered **x** to the star system numbered **y**, thereby ending up **t** ($-1000 \leq t \leq 1000$) years in the future or past, a negative integer denotes past, positive integer denotes future.

Output

For each case, print the case number first. Then print the star systems (in ascending order) where she can start her journey. If no such star system is found, print '**impossible**'.

Sample Input	Output for Sample Input
<pre>2 3 3 0 1 1000 1 2 15 2 1 -42 4 4 0 1 10 1 2 20 2 3 30 3 0 -60</pre>	<pre>Case 1: 0 1 2 Case 2: impossible</pre>

1109 - False Ordering

We define **b** is a Divisor of a number **a** if **a** is divisible by **b**. So, the divisors of 12 are 1, 2, 3, 4, 6, 12. So, 12 has 6 divisors.

Now you have to order all the integers from 1 to 1000. **x** will come before **y** if

- 1) number of divisors of **x** is less than number of divisors of **y**
- 2) number of divisors of **x** is equal to number of divisors of **y** and **x > y**.

Input

Input starts with an integer **T** (≤ 1005), denoting the number of test cases.

Each case contains an integer **n** ($1 \leq n \leq 1000$).

Output

For each case, print the case number and the **nth** number after ordering.

Sample Input	Output for Sample Input
5	Case 1: 1
1	Case 2: 997
2	Case 3: 991
3	Case 4: 983
4	Case 5: 840
1000	

1110 - An Easy LCS

LCS means 'Longest Common Subsequence' that means two non-empty strings are given; you have to find the Longest Common Subsequence between them. Since there can be many solutions, you have to print the one which is the lexicographically smallest. Lexicographical order means dictionary order. For example, 'abc' comes before 'abd' but 'aaz' comes before 'abc'.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a blank line. The next two lines will contain two strings of length **1** to **100**. The strings contain lowercase English characters only.

Output

For each case, print the case number and the lexicographically smallest LCS. If the LCS length is **0** then just print ':('.

Sample Input	Output for Sample Input
3 ab ba zxcvbn hjgasbznxbzmx you kjhs	Case 1: a Case 2: zxb Case 3: :(

1111 - Best Picnic Ever

K people are having a picnic. They are initially in **N** cities, conveniently numbered from **1** to **N**. The roads between cities are connected by **M** one-way roads (no road connects a city to itself).

Now they want to gather in the same city for their picnic, but (because of the one-way roads) some people may only be able to get to some cities. Help them by figuring out how many cities are reachable by all of them, and hence are possible picnic locations.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with three integers **K** ($1 \leq K \leq 100$), **N** ($1 \leq N \leq 1000$), **M** ($1 \leq M \leq 10000$). Each of the next **K** lines will contain an integer (**1** to **N**) denoting the city where the **ith** person lives. Each of the next **M** lines will contain two integers **u v** ($1 \leq u, v \leq N, u \neq v$) denoting there is a road from **u** to **v**.

Output

For each case, print the case number and the number of cities that are reachable by all of them via the one-way roads.

Sample Input	Output for Sample Input
1 2 4 4 2 3 1 2 1 4 2 3 3 4	Case 1: 2

1112 – Curious Robin Hood

Robin Hood likes to loot rich people since he helps the poor people with this money. Instead of keeping all the money together he does another trick. He keeps **n** sacks where he keeps this money. The sacks are numbered from **0** to **n-1**.

Now each time he can do one of the three tasks.

- 1) Give all the money of the i^{th} sack to the poor, leaving the sack empty.
- 2) Add new amount (given in input) in the i^{th} sack.
- 3) Find the total amount of money from i^{th} sack to j^{th} sack.

Since he is not a programmer, he seeks your help.

Input

Input starts with an integer **T (≤ 5)**, denoting the number of test cases.

Each case contains two integers **n ($1 \leq n \leq 10^5$)** and **q ($1 \leq q \leq 50000$)**. The next line contains **n** space separated integers in the range **[0, 1000]**. The i^{th} integer denotes the initial amount of money in the i^{th} sack (**$0 \leq i < n$**).

Each of the next **q** lines contains a task in one of the following form:

- 1 i** Give all the money of the i^{th} (**$0 \leq i < n$**) sack to the poor.
- 2 i v** Add money **v ($1 \leq v \leq 1000$)** to the i^{th} (**$0 \leq i < n$**) sack.
- 3 i j** Find the total amount of money from i^{th} sack to j^{th} sack (**$0 \leq i \leq j < n$**).

Output

For each test case, print the case number first. If the query type is **1**, then print the amount of money given to the poor. If the query type is **3**, print the total amount from i^{th} to j^{th} sack.

Sample Input	Output for Sample Input
1 5 6 3 2 1 4 5 1 4 2 3 4 3 0 3 1 2 3 0 4 1 1	Case 1: 5 14 1 13 2

Notes

Dataset is huge, use faster I/O methods.

1113 - Discover the Web

Standard web browsers contain features to move backward and forward among the pages recently visited. One way to implement these features is to use two stacks to keep track of the pages that can be reached by moving backward and forward. You are asked to implement this. The commands are:

1. BACK: If the backward stack is empty, the command is ignored. Otherwise, push the current page on the top of the forward stack. Pop the page from the top of the backward stack, making it the new current page.
2. FORWARD: If the forward stack is empty, the command is ignored. Otherwise, push the current page on the top of the backward stack. Pop the page from the top of the forward stack, making it the new current page.
3. VISIT <url>: Push the current page on the top of the backward stack, and make the URL specified the new current page. The forward stack is emptied.
4. QUIT: Quit the browser.

The browser initially loads the web page at the URL '<http://www.lightoj.com/>'

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains some commands. The keywords BACK, FORWARD, VISIT, and QUIT are all in uppercase. URLs have no whitespace and have at most 50 characters. The end of case is indicated by the QUIT command and it shouldn't be processed. Each case contains at most 100 lines.

Output

For each case, print the case number first. For each command, print the URL of the current page (in a line) after the command is executed if the command is not ignored. Otherwise, print '**Ignored**'.

Sample Input	Output for Sample Input
1 VISIT http://uva.onlinejudge.org/ VISIT http://topcoder.com/ BACK BACK BACK FORWARD VISIT http://acm.sgu.ru/ BACK BACK FORWARD FORWARD FORWARD QUIT	Case 1: http://uva.onlinejudge.org/ http://topcoder.com/ http://uva.onlinejudge.org/ http://www.lightoj.com/ Ignored http://uva.onlinejudge.org/ http://acm.sgu.ru/ http://uva.onlinejudge.org/ http://www.lightoj.com/ http://uva.onlinejudge.org/ http://acm.sgu.ru/ Ignored

1114 – Easily Readable

As you probably know, the human information processor is a wonderful text recognizer that can handle even sentences that are garbled like the following:

The ACM Itrenntaoial Clloegaite Porgarmmnig Cnotset (IPCC) porvdies colgee stuetnds wtih ooppriuntetiis to itnrecat wtih sutednts form ohetr uinevsrtieis.

People have claimed that understanding these sentences works in general when using the following rule: The first and last letters of each word remain unmodified and all the characters in the middle can be reordered freely. Since you are an ACM programmer, you immediately set on to write the following program: Given a sentence and a dictionary of words, how many different sentences can you find that could potentially be mapped to the same encoding?

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing the number **n ($0 \leq n \leq 10000$)** of words in the dictionary, which are printed on the following **n** lines. After this, there is a line containing the number **m ($0 \leq m \leq 10000$)** of sentences that should be tested with the preceding dictionary and then **m** lines containing those sentences. The sentences consist of letters from **a** to **z**, **A** to **Z** and spaces only and have a maximal length of **10000** characters. For each word in the dictionary a limitation of **100** characters can be assumed. The words are case sensitive. In any case, total number of characters in the sentences will be at most **10^5** . And total characters in the dictionary will be at most **10^5** .

Output

For each case, print the case number first. Then for each sentence, output the number of sentences that can be formed on an individual line. Result fits into 32 bit signed integer.

Sample Input	Output for Sample Input
1 8 baggers beggars in the blowed bowled barn bran 1 beggars bowled in the barn	Case 1: 8

Note

Dataset is huge, use faster I/O methods.

1115 – Filling the Regions

Given a model of a map in a 2D grid, you have to color the map satisfying certain constraints.

- 1) A map contains one or more regions. Each region is identified by an uppercase English letter L_i . In the grid, there will be some cells containing L_i to form the region. From any cell of that region, it's possible to go to all cells (in that region) using adjacent moves. Adjacent move means going to a cell which shares a side with the current cell and belongs to the same region.
- 2) A region Q may be surrounded by another region P . In such case, Q is called a sub-region of P and Q may be erased from the map. Mark all the cells that are surrounded by P with P 's identifier letter.
- 3) If two cells of a region share a corner, then there will always be at least one cell which shares sides with both the cells.

Now your job is to report the updated map after filling the regions.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with two integers m and n ($5 \leq m, n \leq 50$) denoting the number of rows and columns respectively. Each of the next m lines contains n characters each. Each character will be either a '!' or any uppercase English letter. '!' means empty place, and letters represent regions as described above. You can assume that the input data satisfies the above constraints.

Output

For each case, print the case number first. Print m lines, each line with n characters showing the final grid after erasing sub-regions and filling the surrounding cells.

Sample Input	Output for Sample Input
2 5 5 AAAEE ABAHE A.AHF AAAHF .G... 20 20B..... ..BBBB..... ..B..BBB..... .BB....B..... BBB..RRBBBBBBBBB... BBB..RR....SS.BBB... .BB..RR....SSBB..... ..B..RR.DD.BBB..... .BB..KMDDD.B..... ..BB.KDD...B..BBB... ..B.KK....BBBBBB... ..BB.KK....BBCB... ..B.K..BB..BCBB... ...BBBBBB....BBBB... ...BBBBBBB....BBB...BB...BB..B..	Case 1: AAAEE AAAHE AAAHF AAAHF .G... Case 2:B..... ..BBBB..... ..BBBBBB..... ..BBBBBBB..... ..BBBBBBB... BBBBBBBBBBBBBBBBB... BBBBBBBBBBBBBBBBB... ..BBBBBBBBBBBBB... ..BBBBBBBBBBBB... ..BBBBBBBBBBB... ..BBBBBBBBBBB... ..BBBBBBBBBBB... ..BBBBBBBB... ..BBBBBBB... ..BBBBBB... ..BBBBBB... ..BBBBBB... ..BBBBBB... ...BB...BB..B...

1116 – Ekka Dokka

Ekka and his friend Dokka decided to buy a cake. They both love cakes and that's why they want to share the cake after buying it. As the name suggested that Ekka is very fond of odd numbers and Dokka is very fond of even numbers, they want to divide the cake such that Ekka gets a share of **N** square centimeters and Dokka gets a share of **M** square centimeters where **N** is odd and **M** is even. Both **N** and **M** are positive integers.

They want to divide the cake such that **N * M = W**, where **W** is the dashing factor set by them. Now you know their dashing factor, you have to find whether they can buy the desired cake or not.

Input

Input starts with an integer **T (≤ 10000)**, denoting the number of test cases.

Each case contains an integer **W ($2 \leq W < 2^{63}$)**. And **W** will not be a power of **2**.

Output

For each case, print the case number first. After that print "**Impossible**" if they can't buy their desired cake. If they can buy such a cake, you have to print **N** and **M**. If there are multiple solutions, then print the result where **M** is as small as possible.

Sample Input	Output for Sample Input
3 10 5 12	Case 1: 5 2 Case 2: Impossible Case 3: 3 4

1117 - Helping Cicada

Cicada is an insect with large transparent eyes and well-veined wings similar to the "jar flies". The insects are thought to have evolved 1.8 million years ago during the Pleistocene epoch. There are about 2,500 species of cicada around the world which live in temperate tropical climates.

These are all sucking insects, which pierce plants with their pointy mouthparts and suck out the juices. But there are some predators (like birds, the Cicada Killer Wasp) that attack cicadas. Each of the predators has a periodic cycle of attacking Cicadas. For example, birds attack them every three years; wasps attack them every 2 years. So, if Cicadas come in the 12th year, then birds or wasps can attack them. If they come out in the 7th year then no one will attack them.



So, at first they will choose a number **N** which represents possible life-time. Then there will be an integer **M** indicating the total number of predators. The next **M** integers represent the life-cycle of each predator. The numbers in the range from **1** to **N** which are not divisible by any of those **M** life-cycles numbers will be considered for cicada's safe-emerge year. And you want to help them.

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case contains two integers **N** ($1 \leq N < 2^{31}$) and **M** ($1 \leq M \leq 15$). The next line contains **M** positive integers (fits into 32 bit signed integer) denoting the life cycles of the predators.

Output

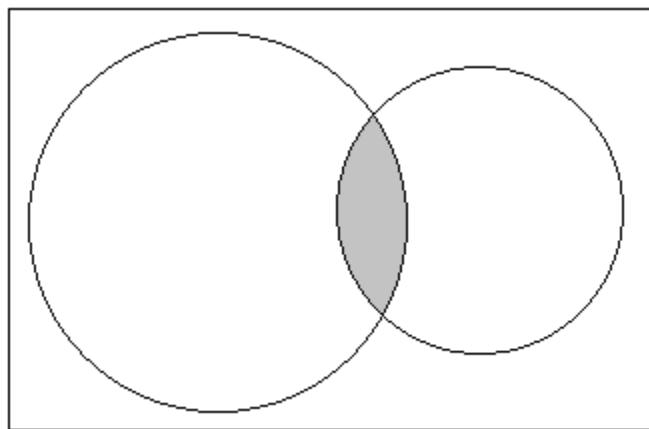
For each test case, print the case number and the number of safe-emerge days for cicada.

Sample Input	Output for Sample Input
2 15 3 2 3 5 10 4 2 4 5 7	Case 1: 4 Case 2: 3

1118 - Incredible Molecules

In the biological lab, you were examining some of the molecules. You got some interesting behavior about some of the molecules. There are some circular molecules, when two of them collide, they overlap with each other, and it's hard to find that which one is over the other one.

Given two molecules as circles, you have to find the common area of the given molecules that is shaded in the picture.



Overlapping Molecules

Input

Input starts with an integer **T (≤ 12)**, denoting the number of test cases.

Each case contains six integers x_1, y_1, r_1 and x_2, y_2, r_2 . Where (x_1, y_1) is the center of the first molecule and r_1 is the radius and (x_2, y_2) is the center of the second molecule and r_2 is the radius. Both the radiuses are positive. No integer will contain more than **3** digits.

Output

For each test case, print the case number and the common area of the given molecules. Errors less than **10^{-6}** will be ignored.

Sample Input	Output for Sample Input
3 0 0 10 15 0 10 -10 -10 5 0 -10 10 100 100 20 100 110 20	Case 1: 45.3311753978 Case 2: 35.07666099 Case 3: 860.84369

1119 – Pimp My Ride

Today, there are quite a few cars, motorcycles, trucks and other vehicles out there on the streets that would seriously need some refurbishment. You have taken on this job, ripping off a few dollars from a major TV station along the way. Of course, there's a lot of work to do, and you have decided that it's getting too much. Therefore you want to have the various jobs like painting, interior decoration and so on done by garages. Unfortunately, those garages are very specialized, so you need different garages for different jobs. More so, they tend to charge you the more the better the overall appearance of the car is. That is, a painter might charge more for a car whose interior is all leather. As those "surcharges" depend on what job is done and which jobs have been done before, you are currently trying to save money by finding an optimal order for those jobs.

Individual jobs are numbered **1** through **n**. Given the base price **p** for each job and a surcharge **s** for every pair of jobs **(i, j)**, meaning that you have to pay additional **s** for job **i**, if and only if job **j** was completed before, you are to compute the minimum total costs needed to finish all jobs.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with an integer **n** ($1 \leq n \leq 14$) denoting number of jobs. Then follow **n** lines, each containing exactly **n** integers. The **ith** line contains the surcharges that have to be paid in garage number **i** for the **ith** job and the base price for job **i**. More precisely, on the **ith** line, the **ith** integer is the base price for job **i** and the **jth** integer **i ≠ j** is the surcharge for job **i** that applies if job **j** has been done before. The prices will be non-negative integers smaller than or equal to **100000**.

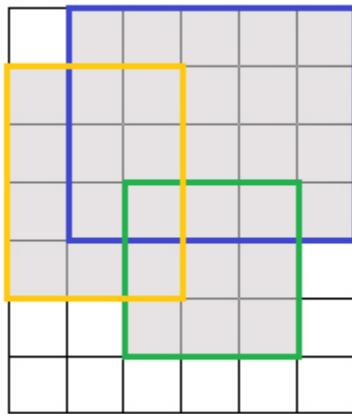
Output

For each case, print the case number and the minimum total cost.

Sample Input	Output for Sample Input
2 2 10 10 9000 10 3 14 23 0 0 14 0 1000 9500 14	Case 1: 30 Case 2: 42

1120 - Rectangle Union

Given some axis parallel rectangles, you have to find the union of their area. For example, see the shaded regions in the picture. Each rectangle will be denoted by four integers. They are x_1, y_1, x_2, y_2 where (x_1, y_1) denotes the lower left corner and (x_2, y_2) denotes the upper right corner.



For the picture above, there are three rectangles. For the yellow rectangle the co-ordinates are $(0, 2)$ and $(3, 6)$. For the blue rectangle the co-ordinates are $(1, 3)$ and $(6, 7)$. For the green rectangle the co-ordinates are $(2, 1)$ and $(5, 4)$. So, the union area is (the shaded region) 31 square units.

Input

Input starts with an integer T (≤ 13), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 30000$). Each of the next n lines will contain four integers x_1, y_1, x_2, y_2 ($0 \leq x_1, y_1, x_2, y_2 \leq 10^9, x_1 < x_2, y_1 < y_2$) denoting a rectangle.

Output

For each case, print the case number and the union area.

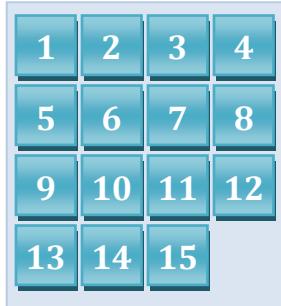
Sample Input	Output for Sample Input
2 3 0 2 3 6 1 3 6 7 2 1 5 4 2 0 0 4 4 1 1 2 5	Case 1: 31 Case 2: 17

Notes

Dataset is huge, use faster I/O methods.

1121 - 15 Puzzle

The 15-puzzle is a very popular game; even if you don't know it by that name, you've seen it. It is constructed with 15 sliding tiles, each with a number from 1 to 15 on it, and all packed into a 4 by 4 frame with one tile missing. The objective of the puzzle is to arrange the tiles so that they are ordered as below:



Here the only legal operation is to exchange missing tile with one of the tiles with which it shares an edge. As an example, the following sequence of moves changes the status of a puzzle



A random puzzle Position



The missing Tile
moves right. Denoted
by **R**



The missing Tile
moves left. Denoted
by **L**



The missing Tile
moves upwards.
Denoted by **U**



The missing Tile
moves down. Denoted
by **D**

The letters in the previous row indicate which neighbor of the missing tile is swapped with it at each step; legal values are 'R','L','U' and 'D', for RIGHT, LEFT, UP, and DOWN, respectively.

Given an initial configuration of a 15-puzzle you will have to determine the steps that would make you reach the final stage. You have to find the minimum number of steps to solve the puzzle. If there are several solutions print the one that comes lexicographically earlier. If there is no solution or you need more than 35 moves; report so.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains a blank line and 16 integers in 4 lines representing the initial board. 0 denotes the blank. You may assume that every integer from 0 to 15 occurs exactly once in a board. And you may also assume that you need at least one move to solve the puzzle.

Output

For each case of input, print the case number first. Then if there is no solution, or you need more than 35 moves, print '**This puzzle is not solvable.**' Otherwise print the steps in a line. If several solutions are there taking minimum steps, print the one that is lexicographically smallest.

Sample Input	Output for Sample Input
2 2 3 4 0 1 5 7 8 9 6 10 12 13 14 11 15 13 1 2 4 5 0 3 7 9 6 10 12 15 8 11 14	Case 1: LLLDRDRDR Case 2: This puzzle is not solvable.

1122 - Digit Count

Given a set of digits **S**, and an integer **n**, you have to find how many **n**-digit integers are there, which contain digits that belong to **S** and the difference between any two adjacent digits is not more than two.

Input

Input starts with an integer **T** (≤ 300), denoting the number of test cases.

Each case contains two integers, **m** ($1 \leq m < 10$) and **n** ($1 \leq n \leq 10$). The next line will contain **m** integers (**from 1 to 9**) separated by spaces. These integers form the set **S** as described above. These integers will be distinct and given in ascending order.

Output

For each case, print the case number and the number of valid **n**-digit integers in a single line.

Sample Input	Output for Sample Input
3 3 2 1 3 6 3 2 1 2 3 3 3 1 4 6	Case 1: 5 Case 2: 9 Case 3: 9

Note

For the first case the valid integers are

11
13
31
33
66

1123 – Trail Maintenance

Tigers in the Sunderbans wish to travel freely among the **N** fields (numbered from **1** to **N**), even though they are separated by trees. The tigers wish to maintain trails between pairs of fields so that they can travel from any field to any other field using the maintained trails. Tigers may travel along a maintained trail in either direction.

The tigers do not build trails. Instead, they maintain deer trails that they have discovered. On any week, they can choose to maintain any or all of the deer animal trails they know about. Always curious, the tigers discover one new deer trail at the beginning of each week. They must then decide the set of trails to maintain for that week so that they can travel from any field to any other field. Tigers can only use trails which they are currently maintaining.

The tigers always want to minimize the total length of trail they must maintain. The tigers can choose to maintain any subset of the deer trails they know about, regardless of which trails were maintained the previous week. Deer trails (even when maintained) are never straight. Two trails that connect the same two fields might have different lengths. While two trails might cross, tigers are so focused; they refuse to switch trails except when they are in a field. At the beginning of each week, the tigers will describe the deer trail they discovered. Your program must then output the minimum total length of trail the tigers must maintain that week so that they can travel from any field to any other field, if there is such a set of trails.

Input

Input starts with an integer **T** (≤ 25), denoting the number of test cases.

Each case starts with two integers **N** ($1 \leq N \leq 200$) and **W**. **W** is the number of weeks the program will cover ($1 \leq W \leq 6000$).

Each of the next **W** lines will contain three integers describing the trail the tigers found that week. The first two numbers denote the end points (field numbers) and the third number denotes the length of the trail (**1 to 10000**). No trail has the same field as both of its end points.

Output

For each case, print the case number in a line. Then for every week, output a single line with the minimum total length of trail the tigers must maintain so that they can travel from any field to any other field. If no set of trails allows the tigers to travel from any field to any other field, output "**-1**".

Sample Input	Output for Sample Input
1 4 6 1 2 10 1 3 8 3 2 3 1 4 3 1 3 6 2 1 2	Case 1: -1 -1 -1 14 12 8

1124 - Cricket Ranking

World Cup Cricket 2011 is coming. Many of you may not be interested in cricket, but it's really a passion in our subcontinent. Ranking of cricketers is a common phenomenon there. Cricketers are ranked by their performance in both form of cricket (Test and One-day). People really enjoy this ranking. They like to see their favorite players as top ranked. Currently many such rankings are available. And as you have guessed, each ranking makes a different player as top ranked.

World Cup committee has decided that they will make a new ranking on the performance of players in the world cup. They want the ranking to be acceptable to the public. The rules are:

1. There are **K** different departments. Cricketers will be given points in each department depending on their performance.
2. The maximum points for each department are not equal. Such as Sakib Al Hasan can get maximum 25 points in batting but Mushfiqur Rahim can get maximum 10 points for his spectacular wicket keeping.
3. The sum of maximum points of all departments will be exactly **N** points. And the final ranking will depend on the total earned points out of **N** points.

The ranking committee wants popular cricketers to get top ranked. To do so, they even allow maximum points for fielding more than that of batting. But that can bring lots of criticism. So they decide to fix a range of points for each department. Such as maximum points for batting will be at least 10 and at most 15 or for fielding, it will be at least 8 and at most 12. But the total points will be 20. Then 3 ranking systems are possible, such as: (10, 10), (11, 9) and (12, 8) for batting and fielding respectively. In this problem, you have to find the number of ranking systems possible for given number of departments, range of points for each department and total points.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a blank line and two integers **K** ($1 \leq K \leq 10$) and **N** ($1 \leq N \leq 10^6$). Each of the next **K** lines denotes the lower and upper limit of allowable maximum points for each department. These integers will be in the range $[0, 10^6]$. And the lower limit of each department will always be less than or equal to the upper limit of that department.

Output

For each case, print the case number and the number of rankings possible modulo **100000007**.

Sample Input	Output for Sample Input
<pre> 4 4 10 1 1 2 2 3 3 4 4 4 10 1 1 2 2 3 3 3 3 2 10 1 10 1 10 2 20 10 15 8 12 </pre>	<pre> Case 1: 1 Case 2: 0 Case 3: 9 Case 4: 3 </pre>

1125 - Divisible Group Sums

Given a list of **N** numbers you will be allowed to choose any **M** of them. So you can choose in ${}^N C_M$ ways. You will have to determine how many of these chosen groups have a sum, which is divisible by **D**.

Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

The first line of each case contains two integers **N** ($0 < N \leq 200$) and **Q** ($0 < Q \leq 10$). Here **N** indicates how many numbers are there and **Q** is the total number of queries. Each of the next **N** lines contains one 32 bit signed integer. The queries will have to be answered based on these **N** numbers. Each of the next **Q** lines contains two integers **D** ($0 < D \leq 20$) and **M** ($0 < M \leq 10$).

Output

For each case, print the case number in a line. Then for each query, print the number of desired groups in a single line.

Sample Input	Output for Sample Input
2 10 2 1 2 3 4 5 6 7 8 9 10 5 1 5 2 5 1 2 3 4 5 6 6 2	Case 1: 2 9 Case 2: 1

1126 – Building Twin Towers

Professor Sofdor Ali is fascinated about twin towers. So, in this problem you are working as his assistant, and you have to help him making a large twin towers.

For this reason he gave you some rectangular bricks. You can pick some of the bricks and can put bricks on top of each other to build a tower. As the name says, you want to make two towers that have equal heights. And of course the height of the towers should be positive.

For example, suppose there are three bricks of heights 3, 4 and 7. So you can build two towers of height 7. One contains a single brick of height 7, and the other contains a brick of height 3 and a brick of height 4. If you are given bricks of heights 2, 2, 3 and 4 then you can make two towers with height 4 (just don't use brick with height 3).

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with an integer **n ($1 \leq n \leq 50$)**, denoting the number of bricks. The next line contains **n** space separated integers, each containing the height of the brick **h_i ($1 \leq h_i \leq 500000$)**. You can safely assume that the sum of heights of all bricks will not be greater than **500000**.

Output

For each case, print the case number and the height of the tallest twin towers that can be built. If it's impossible to build the twin towers as stated, print "**impossible**".

Sample Input	Output for Sample Input
4 3 3 4 7 3 10 9 2 2 21 21 9 15 15 14 24 14 3 20 23 15	Case 1: 7 Case 2: impossible Case 3: 21 Case 4: 64

1127 - Funny Knapsack

Given n integers and a knapsack of weight W , you have to count the number of combinations for which you can add the items in the knapsack without overflowing the weight.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case contains two integers n ($1 \leq n \leq 30$) and W ($1 \leq W \leq 2 * 10^9$) and the next line will contain n integers separated by spaces. The integers will be non negative and less than 10^9 .

Output

For each set of input, print the case number and the number of possible combinations.

Sample Input	Output for Sample Input
3 1 1 1 1 1 2 3 10 1 2 4	Case 1: 2 Case 2: 1 Case 3: 8

1128 – Greatest Parent

A tree is a connected graph with no cycles. In this problem you are given a rooted tree, where each node contains an integer value. And the value of a node is strictly greater than the value of its parent. Now you are given a node and an integer query. You have to find the greatest possible parent of this node (may include the node itself), whose value is greater than or equal to the given query integer.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

The first line of a case is a blank line. The next line contains two integers **N** ($1 \leq N \leq 10^5$), **q** ($1 \leq q \leq 50000$) where **N** denotes the number of nodes and **q** denotes the number of queries. The nodes are numbered from **0** to **N-1**.

Then there will be **N-1** lines. The **ith** ($1 \leq i < N$) line contains two integers **p_i** and **s_i** ($0 \leq p_i < i$, $1 \leq s_i < 2^{31}$). **p_i** denotes the parent and **s_i** denotes the value of the **ith** node, respectively. Assume that the given tree is correct and follows the restrictions as stated. You can assume that the **0th** node is the root and its value is 1.

Each of the next **q** lines contains a query. Each query contains two integers: **k** and **v** ($1 \leq k < N$, $1 \leq v \leq s_k$).

Output

For each case, print the case number in a line. Then for each query, print the index of the greatest parent of node **k** with value greater than or equal to **v**. You can assume that a solution exists.

Sample Input	Output for Sample Input
1 7 4 0 3 0 4 0 2 1 4 2 7 2 10 5 1 4 2 5 4 6 10	Case 1: 0 1 2 6

Note

Dataset is huge. Use faster I/O methods.

1129 – Consistency Checker

SETI is receiving some weird signals for the past few days. After converting them to our number system they found out that some numbers are repeating. Due to travelling millions of miles signal gets distorted. Now they asked you check the consistency of their data sets. The consistency is that, no number is the prefix another number in that data set. Let's consider a data set:

1. 123
2. 5123456
3. 123456

In this data set, numbers not consistent, because the first number is the prefix of the third one.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with an integer **n ($1 \leq n \leq 10000$)** denoting the total numbers in their list. Each of the next **n** lines contains one unique phone number. A number is a sequence of digits and has length from **1** to **10**.

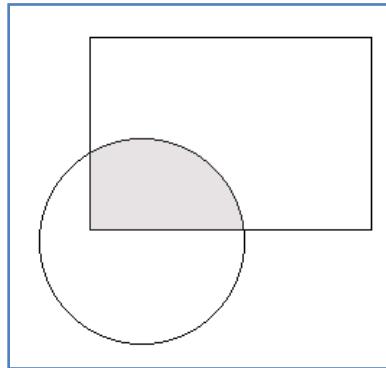
Output

For each case, print the case number and '**YES**' if the list is consistent or '**NO**' if it's not.

Sample Input	Output for Sample Input
2 3 911 97625999 91125426 5 113 12340 123440 12345 98346	Case 1: NO Case 2: YES

1130 - Intersection between Circle and Rectangle

Given the co-ordinates of a circle and the lower left and upper right coordinate of an axis parallel rectangle, you have to find their common area. In the picture, the shaded area is their common area.



Input

Input starts with an integer **T** (≤ 4000), denoting the number of test cases.

Each case contains two lines, first line contains three integers **x, y, r** where **(x, y)** denotes the center of the circle and **r** denotes the radius. The next line contains four integers **x₁, y₁, x₂, y₂** ($x_1 < x_2$, $y_1 < y_2$). All the integers are non-negative and not greater than **200**.

Output

For each case, print the case number their common area. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3 1 1 10 2 2 5 5 1 1 10 20 20 30 30 1 1 5 6 1 8 8	Case 1: 9 Case 2: 0 Case 3: 24.6014178376

1131 – Just Two Functions

Let

$$f_n = a_1 * f_{n-1} + b_1 * f_{n-2} + c_1 * g_{n-3}$$

$$g_n = a_2 * g_{n-1} + b_2 * g_{n-2} + c_2 * f_{n-3}$$

Find $f_n \% M$ and $g_n \% M$. ($\%$ stands for the modulo operation.)

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers $a_1 \ b_1 \ c_1$ ($0 \leq a_1, b_1, c_1 < 25000$). Next line contains three integers $a_2 \ b_2 \ c_2$ ($0 \leq a_2, b_2, c_2 < 25000$). Next line contains three integers $f_0 \ f_1 \ f_2$ ($0 \leq f_0, f_1, f_2 < 25000$). Next line contains three integers $g_0 \ g_1 \ g_2$ ($0 \leq g_0, g_1, g_2 < 25000$). The next line contains an integer M ($1 \leq M < 25000$).

Next line contains an integer q ($1 \leq q \leq 100$) denoting the number of queries. Next line contains q space separated integers denoting n . Each of these integers is non-negative and less than 2^{31} .

Output

For each case, print the case number in a line. Then for each query, you have to print one line containing $f_n \% M$ and $g_n \% M$.

Sample Input	Output for Sample Input
2 1 1 0 0 0 0 0 1 1 0 0 0 20000 10 1 2 3 4 5 6 7 8 9 10 1 1 1 1 1 1 2 2 2 2 2 2 20000 5 2 4 6 8 10	Case 1: 1 0 1 0 2 0 3 0 5 0 8 0 13 0 21 0 34 0 55 0 Case 2: 2 2 10 10 34 34 114 114 386 386

1132 – Summing up Powers

Given \mathbf{N} and \mathbf{K} , you have to find

$$(1^K + 2^K + 3^K + \dots + N^K) \% 2^{32}$$

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case contains two integers N ($1 \leq N \leq 10^{15}$) and K ($0 \leq K \leq 50$) in a single line.

Output

For each case, print the case number and the result.

Sample Input	Output for Sample Input
3 3 1 4 2 3 3	Case 1: 6 Case 2: 30 Case 3: 36

1133 – Array Simulation

I am retired now, so, no work, a lot of time to spare and a lot of problems to share. Well, finally I am thinking of the old days when I was a solver. But now I am stuck with a tough problem that I want to share with you.

Given an array and some operations on the array, you have to print the final state of the array. Say, the array is $a[]$, the size is n and indexed from **0** to **n-1**.

The operations are:

1. '**S D**'. **D** is an integer. **D** will be added with all the elements of the array.
2. '**M D**'. **D** is an integer. All the elements of the array will be multiplied by **D**.
3. '**D K**'. **K** is a non zero integer. All the elements of the array will be divided by **K** (integer division).
4. '**R**'. It means reverse. It will reverse the elements of the array.
5. '**P Y Z**'. **Y** and **Z** are integers. It will swap the elements $a[Y]$ and $a[Z]$.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains two integers **n** ($1 \leq n \leq 100$) and **m** ($0 \leq m \leq 101$). The next line contains **n** space separated integers denoting the elements of the array. Each of the next **m** lines contains an operation defined above. You can assume that no operation will overflow/underflow the 32 bit signed integer range or access any invalid array reference.

Output

For each case, print the case number first. In the next line you have to print the elements of the array. Two elements should be separated by a single space. There should be no trailing space after the last integer of the array.

Sample Input	Output for Sample Input
2 5 3 1 2 3 4 5 P 0 1 S 1 R 4 2 2 7 8 1 M 10 D 5	Case 1: 6 5 4 2 3 Case 2: 4 14 16 2

1134 - Be Efficient

You are given an array with **N** integers, and another integer **M**. You have to find the number of consecutive subsequences which are divisible by **M**.

For example, let **N** = 4, the array contains {2, 1, 4, 3} and **M** = 4.

The consecutive subsequences are {2}, {2 1}, {2 1 4}, {2 1 4 3}, {1}, {1 4}, {1 4 3}, {4}, {4 3} and {3}. Of these 10 'consecutive subsequences', only two of them adds up to a figure that is a multiple of 4 - {1 4 3} and {4}.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case contains two integers **N** ($1 \leq N \leq 10^5$) and **M** ($1 \leq M \leq 10^5$). The next line contains **N** space separated integers forming the array. Each of these integers will lie in the range [1, 10^5].

Output

For each case, print the case number and the total number of consecutive subsequences that are divisible by **M**.

Sample Input	Output for Sample Input
2 4 4 2 1 4 3 6 3 1 2 3 4 5 6	Case 1: 2 Case 2: 11

Note

Dataset is huge. Use faster i/o methods.

1135 – Count the Multiples of 3

You have an array with **n** elements which is indexed from **0** to **n - 1**. Initially all elements are zero. Now you have to deal with two types of operations

1. Increase the numbers between indices **i** and **j** (inclusive) by **1**. This is represented by the command '**0 i j**'.
2. Answer how many numbers between indices **i** and **j** (inclusive) are divisible by **3**. This is represented by the command '**1 i j**'.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

Each case starts with a line containing two integers **n** ($1 \leq n \leq 10^5$) and **q** ($1 \leq q \leq 50000$) denoting the number of queries. Each query will be either in the form '**0 i j**' or '**1 i j**' where **i, j** are integers and $0 \leq i \leq j < n$.

Output

For each case, print the case number first. Then for each query in the form '**1 i j**', print the desired result.

Sample Input	Output for Sample Input
1 10 9 0 0 9 0 3 7 0 1 4 1 1 7 0 2 2 1 2 4 1 8 8 0 5 8 1 6 9	Case 1: 2 3 0 2

Note

Dataset is huge, use faster i/o methods.

1136 - Division by 3

There is sequence 1, 12, 123, 1234, ..., 12345678910, Now you are given two integers **A** and **B**, you have to find the number of integers from **Ath** number to **Bth** (inclusive) number, which are divisible by 3.

For example, let **A** = 3. **B** = 5. So, the numbers in the sequence are, 123, 1234, 12345. And 123, 12345 are divisible by 3. So, the result is 2.

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case contains two integers **A** and **B** ($1 \leq A \leq B < 2^{31}$) in a line.

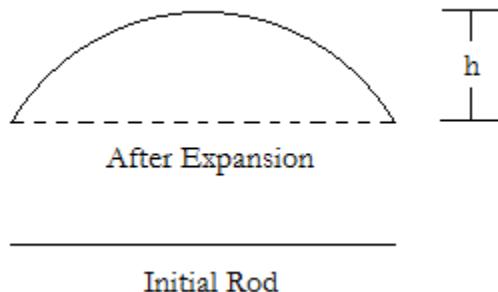
Output

For each case, print the case number and the total numbers in the sequence between **Ath** and **Bth** which are divisible by 3.

Sample Input	Output for Sample Input
2 3 5 10 110	Case 1: 2 Case 2: 67

1137 - Expanding Rods

When a thin rod of length \mathbf{L} is heated \mathbf{n} degrees, it expands to a new length $\mathbf{L}' = (1+n*C)*L$, where \mathbf{C} is the coefficient of heat expansion.



When a thin rod is mounted on two solid walls and then heated, it expands and takes the shape of a circular segment, the original rod being the chord of the segment.

Your task is to compute the distance by which the center of the rod is displaced. That means you have to calculate \mathbf{h} as in the picture.

Input

Input starts with an integer $\mathbf{T} (\leq 20)$, denoting the number of test cases.

Each case contains three non-negative real numbers: the initial length of the rod in millimeters \mathbf{L} , the temperature change in degrees \mathbf{n} and the coefficient of heat expansion of the material \mathbf{C} . Input data guarantee that no rod expands by more than one half of its original length. All the numbers will be between 0 and 1000 and there can be at most 5 digits after the decimal point.

Output

For each case, print the case number and the displacement of the center of the rod in single line. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3 1000 100 0.0001 150 10 0.00006 10 0 0.001	Case 1: 61.3289915 Case 2: 2.2502024857 Case 3: 0

1138 – Trailing Zeroes (III)

Your task is to find minimal natural number N , so that $N!$ contains exactly Q zeroes on the trail in decimal notation. As you know $N! = 1*2*...*N$. For example, $5! = 120$, 120 contains one zero on the trail.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case contains an integer Q ($1 \leq Q \leq 10^8$) in a line.

Output

For each case, print the case number and N . If no solution is found then print '**impossible**'.

Sample Input	Output for Sample Input
3	Case 1: 5
1	Case 2: 10
2	Case 3: impossible
5	

1139 - 8 Puzzle

An 8 Puzzle is played in a board of 3 by 3. Initially the board is scattered. Your goal is to make your board to the following board:

1 2 3
4 5 6
7 8 0

Here 0 denotes the blank. The moves are described in the following section:

1 2 3	1 2 3	1 2 0	1 0 2	1 6 2
4 0 6	->	4 6 0	->	4 6 3
7 8 5		7 8 5		7 8 5
Initial	right	up	left	down

Input

Input starts with an integer **T (≤ 1000)**, denoting the number of test cases.

Each case starts with a blank line. The next three lines will contain three integers each. You may assume that the given board is valid.

Output

For each case, print the case number and the minimum number of moves required to reach the goal state. If it's impossible, then print '**impossible**'.

Sample Input	Output for Sample Input
3 1 2 3 4 5 6 8 7 0	Case 1: impossible Case 2: 1 Case 3: 18
1 2 3 4 5 6 7 0 8	
1 3 2 4 6 5 7 8 0	

1140 – How Many Zeroes?

Jimmy writes down the decimal representations of all natural numbers between and including m and n , ($m \leq n$). How many zeroes will he write down?

Input

Input starts with an integer T (≤ 11000), denoting the number of test cases.

Each case contains two unsigned 32-bit integers m and n , ($m \leq n$).

Output

For each case, print the case number and the number of zeroes written down by Jimmy.

Sample Input	Output for Sample Input
5 10 11 100 200 0 500 1234567890 2345678901 0 4294967295	Case 1: 1 Case 2: 22 Case 3: 92 Case 4: 987654304 Case 5: 3825876150

1141 - Number Transformation

In this problem, you are given an integer number **s**. You can transform any integer number **A** to another integer number **B** by adding **x** to **A**. This **x** is an integer number which is a prime factor of **A** (please note that 1 and **A** are not being considered as a factor of **A**). Now, your task is to find the minimum number of transformations required to transform **s** to another integer number **t**.

Input

Input starts with an integer **T** (≤ 500), denoting the number of test cases.

Each case contains two integers: **s** ($1 \leq s \leq 100$) and **t** ($1 \leq t \leq 1000$).

Output

For each case, print the case number and the minimum number of transformations needed. If it's impossible, then print **-1**.

Sample Input	Output for Sample Input
2 6 12 6 13	Case 1: 2 Case 2: -1

1142 – Summing of Powers (II)

Shanto is learning how to power up numbers and he found an efficient way to find k^{th} power of a matrix. He was quite happy with his discovery. Suddenly his sister Natasha came to him and asked him to find the summation of the powers. To be specific his sister gave the following problem.

Let \mathbf{A} be an $n \times n$ matrix. We define $\mathbf{A}^k = \mathbf{A} * \mathbf{A} * \dots * \mathbf{A}$ (k times). Here, $*$ denotes the usual matrix multiplication. You are to write a program that computes the matrix $\mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 + \dots + \mathbf{A}^k$.

Shanto smiled and thought that it would be an easy one. But after a while he found that it's tough for him. Can you help him?

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with two integers n ($1 \leq n \leq 30$) and k ($1 \leq k \leq 10^9$). Each of the next n lines will contain n non-negative integers (not greater than 10).

Output

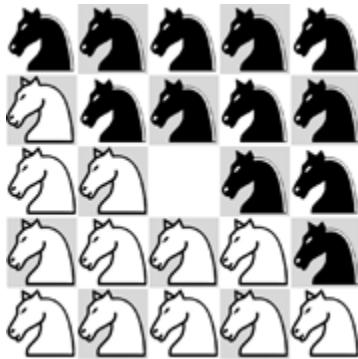
For each case, print the case number and the result matrix. For each cell, just print the last digit. See the samples for more details.

Sample Input	Output for Sample Input
2 3 2 1 4 6 6 5 2 1 2 3 3 10 1 4 6 6 5 2 1 2 3	Case 1: 208 484 722 Case 2: 868 620 546

1143 - Knights in FEN

There are black and white knights on a 5 by 5 chessboard. There are twelve knights of each color, and there is one square that is empty. At any time, a knight can move into an empty square as long as it moves like a knight in normal chess (what else did you expect?).

Given an initial position of the board, the question is: what is the minimum number of moves in which we can reach the final position which is:

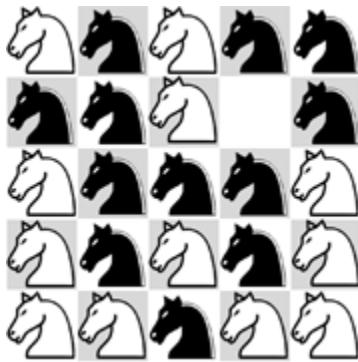


Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains five lines; each line represents one row of a chessboard. The positions occupied by white knights are marked by 0 and the positions occupied by black knights are marked by 1. The space corresponds to the empty square on the board.

The first set of the sample input below corresponds to this configuration:



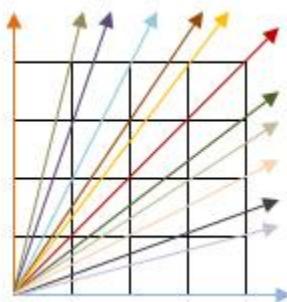
Output

For each case, print the case number and the minimum number of moves leading from the starting input configuration to the final one. If that number is bigger than 15, then output one line stating "**Unsolvable in less than 16 move(s).**" otherwise output one line stating "**Solvable in n move(s).**" where $n \leq 15$.

Sample Input	Output for Sample Input
2 01011 110 1 01110 01010 00100 10110 01 11 10111 01001 00000	Case 1: Unsolvable in less than 16 move(s). Case 2: Solvable in 7 move(s).

1144 - Ray Gun

You are in an $m \times n$ grid. You are standing in position $(0, 0)$ and in each of the other lattice points (points with integer co-ordinates) an enemy is waiting. Now you have a ray gun that can fire up to infinity and no obstacle can stop it. Your target is to kill all the enemies. You have to find the minimum number of times you have to fire to kill all of them. For a 4×4 grid you have to fire **13** times. See the picture below:



Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case contains two integers m, n ($0 \leq m, n \leq 10^9$) and at least one of them will be less than or equal to 10^6 .

Output

For each case, print the case number and the minimum number of times you have to fire to kill all the enemies.

Sample Input	Output for Sample Input
2 4 4 10 10	Case 1: 13 Case 2: 65

1145 – Dice (I)

You have **N** dices; each of them has **K** faces numbered from 1 to **K**. Now you have arranged the **N** dices in a line. You can rotate/flip any dice if you want. How many ways you can set the top faces such that the summation of all the top faces equals **S**?

Now you are given **N**, **K**, **S**; you have to calculate the total number of ways.

Input

Input starts with an integer **T** (≤ 25), denoting the number of test cases.

Each case contains three integers: **N** ($1 \leq N \leq 1000$), **K** ($1 \leq K \leq 1000$) and **S** ($0 \leq S \leq 15000$).

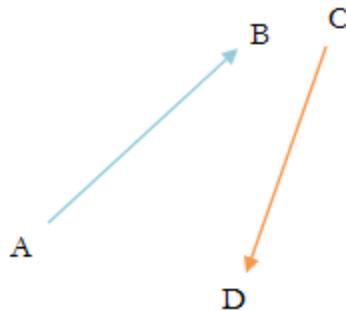
Output

For each case print the case number and the result modulo **100000007**.

Sample Input	Output for Sample Input
5 1 6 3 2 9 8 500 6 1000 800 800 10000 2 100 10	Case 1: 1 Case 2: 7 Case 3: 57286574 Case 4: 72413502 Case 5: 9

1146 - Closest Distance

Two men are moving concurrently, one man is moving from **A** to **B** and other man is moving from **C** to **D**. Initially the first man is at **A**, and the second man is at **C**. They maintain constant velocities such that when the first man reaches **B**, at the same time the second man reaches **D**. You can assume that **A**, **B**, **C** and **D** are 2D Cartesian co-ordinates. You have to find the minimum Euclidean distance between them along their path.



Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case will contain eight integers: $A_x, A_y, B_x, B_y, C_x, C_y, D_x, D_y$. All the co-ordinates are between **0** and **100**. (A_x, A_y) denotes **A**. (B_x, B_y) denotes **B** and so on.

Output

For each case, print the case number and the minimum distance between them along their path. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3 0 0 5 0 5 5 5 0 0 0 5 5 10 10 6 6 0 0 5 0 10 1 1 1	Case 1: 0 Case 2: 1.4142135624 Case 3: 1

1147 - Tug of War

A tug of war is to be arranged at the local office picnic. For the tug of war, the picnickers must be divided into two teams. Each person must be on one team or the other; the number of people on the two teams must not differ by more than 1; the total weight of the people on each team should be as nearly equal as possible.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

The first line of each case is a blank line. The next line of input contains an integer **n** ($2 \leq n \leq 100$), the number of people at the picnic. **n** lines follow. The first line gives the weight of person 1; the second the weight of person 2; and so on. Each weight is an integer between **1** and **100000**. The summation of all the weights of the people in a case will not exceed **100000**.

Output

For each case, print the case number and the total number weights of the people in two teams. If the weights differ, print the smaller weight first.

Sample Input	Output for Sample Input
2 3 100 90 200 4 10 15 17 20	Case 1: 190 200 Case 2: 30 32

1148 – Mad Counting

Mob was hijacked by the mayor of the Town "TruthTown". Mayor wants Mob to count the total population of the town. Now the naive approach to this problem will be counting people one by one. But as we all know Mob is a bit lazy, so he is finding some other approach so that the time will be minimized. Suddenly he found a poll result of that town where N people were asked "How many people in this town other than yourself support the same team as you in the FIFA world CUP 2010?" Now Mob wants to know if he can find the minimum possible population of the town from this statistics. Note that no people were asked the question more than once.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with an integer N ($1 \leq N \leq 50$). The next line will contain N integers denoting the replies (0 to 10^6) of the people.

Output

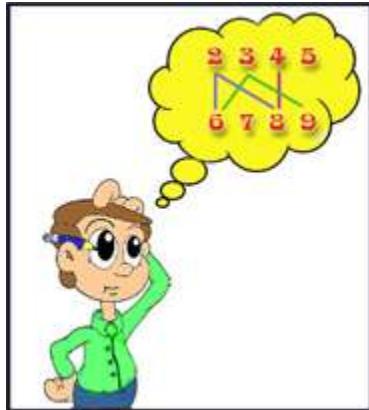
For each case, print the case number and the minimum possible population of the town.

Sample Input	Output for Sample Input
2 4 1 1 2 2 1 0	Case 1: 5 Case 2: 1

1149 – Factors and Multiples

You will be given two sets of integers. Let's call them set **A** and set **B**. Set **A** contains **n** elements and set **B** contains **m** elements. You have to remove **k₁** elements from set **A** and **k₂** elements from set **B** so that of the remaining values no integer in set **B** is a multiple of any integer in set **A**. **k₁** should be in the range **[0, n]** and **k₂** in the range **[0, m]**.

You have to find the value of **(k₁ + k₂)** such that **(k₁ + k₂)** is as low as possible. **P** is a multiple of **Q** if there is some integer **K** such that **P = K * Q**.



Suppose set **A** is **{2, 3, 4, 5}** and set **B** is **{6, 7, 8, 9}**. By removing **2** and **3** from **A** and **8** from **B**, we get the sets **{4, 5}** and **{6, 7, 9}**. Here none of the integers **6, 7** or **9** is a multiple of **4** or **5**.

So for this case the answer is **3** (two from set **A** and one from set **B**).

Input

Input starts with an integer **T** (**≤ 50**), denoting the number of test cases.

The first line of each case starts with an integer **n** followed by **n** positive integers. The second line starts with **m** followed by **m** positive integers. Both **n** and **m** will be in the range **[1, 100]**. Each element of the two sets will fit in a **32** bit signed integer.

Output

For each case of input, print the case number and the result.

Sample Input	Output for Sample Input
2 4 2 3 4 5 4 6 7 8 9 3 100 200 300 1 150	Case 1: 3 Case 2: 0

1150 – Ghosts!

It's a dark, cloudy and spooky night. The ghosts of 'VutPara' graveyard have awakened. They are planning to take revenge against humans. They are dead but the humans are alive that's their main headache. So, they want to frighten all the people nearby.

'Vutpara' can be described as an **n x n** grid. Each cell can be one of the following types:

- '.' - The cell is empty
- 'G' - The cell contains a ghost
- 'H' - The cell contains a human
- '#' - The cell contains over-polluted air, the ghosts can't fly over this cell

The ghosts can move vertically or horizontally but not diagonally. And they can fly to any cell if the air is not over-polluted. It takes one minute to move to an adjacent cell. And it takes two minutes to frighten a human if the ghost is flying over the human (means that the position of the ghost and the human is same). But the ghosts are quite lazy, so any ghost can frighten at most one human. And after their work is done they want to go back to their grave (Initial position).

The night is getting over and they have a little time left. As they are smart enough they know all the human positions and the map. Now they want to frighten all the humans in the map using minimum time.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a blank line and an integer **n ($5 \leq n \leq 25$)**. Then **n** lines follow. Each of the line contains **n** characters each describing 'Vutpara'. You can assume that number of ghosts is always greater than or equal to the number of humans and the number of ghosts is no more than **50**. And there is at least one human in the map.

Output

For each case of input, print the case number and the minimum time needed to frighten all the people or '**Vuter Dol Kupokat**' if it's not possible to frighten all the people.

Sample Input	Output for Sample Input
<p>4</p> <p>8</p> <pre>....##..#.. ..#....G. G....#### H#..HG.G #....#.# H.#G..H. ...##...#</pre>	Case 1: 12 Case 2: 20 Case 3: 12 Case 4: Vuter Dol Kupokat
<p>6</p> <pre>..... G.....H</pre> <p>6</p> <pre>.....G .H.....H. G.....</pre>	
<p>6</p> <pre>#.#G#. G....# G..##. H##.#. ...#H# ..#GHH</pre>	

1151 – Snakes and Ladders

'Snakes and Ladders' or 'Shap-Ludu' is a game commonly played in Bangladesh. The game is so common that it would be tough to find a person who hasn't played it. But those who haven't played it (unlucky of course) the rules are as follows.



1. There is a **10 x 10** board containing some cells numbered from **1** to **100**.
2. You start at position **1**.
3. Each time you throw a perfect dice containing numbers **1** to **6**.
4. There are some snakes and some ladders in the board. Ladders will take you up from one cell to another. Snakes will take you down.
5. If you reach a cell that contains the bottom part of a ladder, you will immediately move to the cell which contains the upper side of that ladder. Similarly if you reach a cell that has a snake-head you immediately go down to the cell where the tail of that snake ends.
6. The board is designed so that from any cell you can jump at most once. (For example there is a snake from 62 to 19, assume that another is from 19 to 2. So, if you reach 62, you will first jump to 19, you will jump to 2. These kinds of cases will not be given)
7. There is no snake head in the 100-th cell and no ladder (bottom part) in the first cell.
8. If you reach cell **100**, the game ends. But if you have to go outside the board in any time your move will be lost. That means you will not take that move and you have to throw the dice again.

Now given a board, you have to find the expected number of times you need to throw the dice to win the game. The cases will be given such that a result will be found.

Input

Input starts with an integer **T (≤ 105)**, denoting the number of test cases.

The first line of a case is a blank line. The next line gives you an integer **n** denoting the number of snakes and ladders. Each of the next **n** lines contain two integers **a** and **b** ($1 \leq a, b \leq 100$, $a \neq b$). If $a < b$, it means that there is a ladder which takes you from **a** to **b**. If $a > b$, it means that there is a snake which takes you from **a** to **b**. Assume that the given board follows the above restrictions.

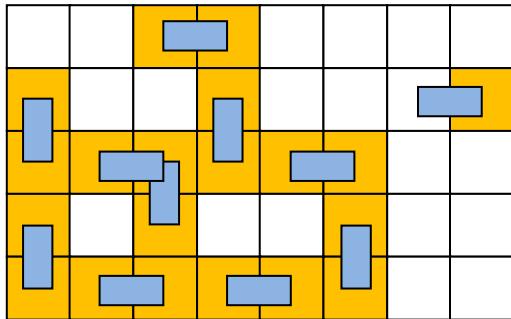
Output

For each case of input, print the case number and the expected number of times you need to throw the dice. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 14 4 42 9 30 16 8 14 77 32 12 37 58 47 26 48 73 62 19 70 89 71 67 80 98 87 24 96 76 0	Case 1: 31.54880806 Case 2: 33.0476190476

1152 - Hiding Gold

You are given a 2D board where in some cells there are gold. You want to fill the board with 2×1 dominoes such that all gold are covered. You may use the dominoes vertically or horizontally and the dominoes may overlap. All you have to do is to cover the gold with least number of dominoes.



In the picture, the golden cells denote that the cells contain gold, and the blue ones denote the 2×1 dominoes. The dominoes may overlap, as we already said, as shown in the picture. In reality the dominoes will cover the full 2×1 cells; we showed small dominoes just to show how to cover the gold with 11 dominoes.

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a row containing two integers m ($1 \leq m \leq 20$) and n ($1 \leq n \leq 20$) and $m * n > 1$. Here m represents the number of rows, and n represents the number of columns. Then there will be m lines, each containing n characters from the set $\{*, o\}$. A '*' character symbolizes the cells which contains a gold, whereas an 'o' character represents empty cells.

Output

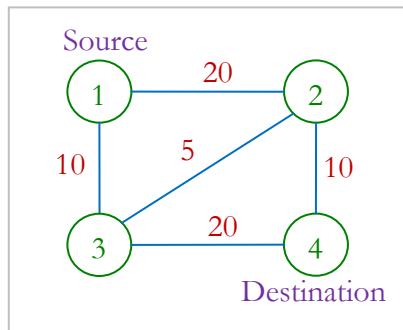
For each case print the case number and the minimum number of dominoes necessary to cover all gold ('*' entries) in the given board.

Sample Input	Output for Sample Input
2 5 8 oo**oooo *ooo*oooo* *****oo *o*oo*oo *****oo 3 4 **oo *oo *oo*	Case 1: 11 Case 2: 4

1153 - Internet Bandwidth

On the Internet, machines (nodes) are richly interconnected, and many paths may exist between a given pair of nodes. The total message-carrying capacity (bandwidth) between two given nodes is the maximal amount of data per unit time that can be transmitted from one node to the other. Using a technique called packet switching; this data can be transmitted along several paths at the same time.

For example, the following figure shows a network with four nodes (shown as circles), with a total of five connections among them. Every connection is labeled with a bandwidth that represents its data-carrying capacity per unit time.



In our example, the bandwidth between node 1 and node 4 is 25, which might be thought of as the sum of the bandwidths 10 along the path 1-2-4, 10 along the path 1-3-4, and 5 along the path 1-2-3-4. No other combination of paths between nodes 1 and 4 provides a larger bandwidth.

You must write a program that computes the bandwidth between two given nodes in a network, given the individual bandwidths of all the connections in the network. In this problem, assume that the bandwidth of a connection is always the same in both directions (which is not necessarily true in the real world).

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Every description starts with a line containing an integer **n ($2 \leq n \leq 100$)**, which is the number of nodes in the network. The nodes are numbered from **1** to **n**. The next line contains three numbers **s**, **t**, and **c**. The numbers **s** and **t** are the source and destination nodes, and the number **c ($c \leq 5000, s \neq t$)** is the total number of connections in the network. Following this are **c** lines describing the connections. Each of these lines contains three integers: the first two are the numbers of the connected nodes, and the third number is the bandwidth of the connection. The bandwidth is a non-negative number not greater than **1000**.

There might be more than one connection between a pair of nodes, but a node cannot be connected to itself. All connections are bi-directional, i.e. data can be transmitted in both directions along a connection, but the sum of the amount of data transmitted in both directions must be less than the bandwidth.

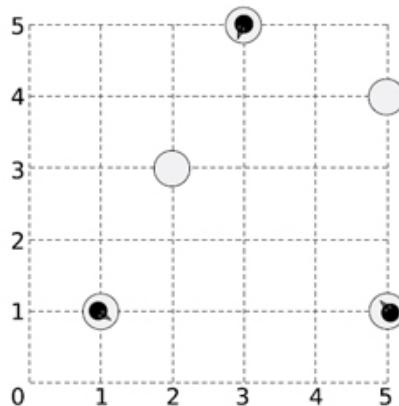
Output

For each case of input, print the case number and the total bandwidth between the source node **s** and the destination node **t**.

Sample Input	Output for Sample Input
2 4 1 4 5 1 2 20 1 3 10 2 3 5 2 4 10 3 4 20 4 1 4 2 1 4 20 1 4 20	Case 1: 25 Case 2: 40

1154 – Penguins

Somewhere near the South Pole, a number of penguins are standing on a number of ice floes. Being social animals, the penguins would like to get together, all on the same floe. The penguins do not want to get wet, so they have used their limited jump distance to get together by jumping from piece to piece. However, temperatures have been high lately, and the floes are showing cracks, and they get damaged further by the force needed to jump to another floe. Fortunately the penguins are real experts on cracking ice floes, and know exactly how many times a penguin can jump off each floe before it disintegrates and disappears. Landing on an ice floe does not damage it. You have to help the penguins find all floes where they can meet.



A sample layout of ice floes with 3 penguins on them

Input

Input starts with an integer T (≤ 25), denoting the number of test cases.

Each case starts with the integer N ($1 \leq N \leq 100$) and a floating-point number D ($0 \leq D \leq 10^5$), denoting the number of ice pieces and the maximum distance a penguin can jump. After that there will be N lines, each line containing x_i , y_i , n_i and m_i , denoting for each ice piece its X and Y coordinate, the number of penguins on it and the maximum number of times a penguin can jump off this piece before it disappears ($-10000 \leq x_i, y_i \leq 10000$, $0 \leq n_i \leq 10$, $1 \leq m_i \leq 200$).

Output

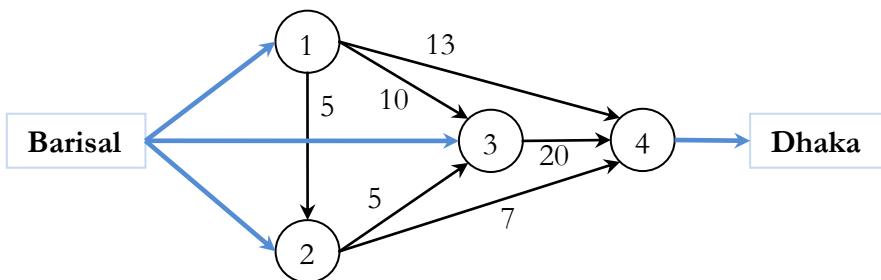
For each case of input, print the case number and a space-separated list of 0-based indices of the pieces on which all penguins can meet. If no such piece exists, output **-1**.

Sample Input	Output for Sample Input
2 5 3.5 1 1 1 1 2 3 0 1 3 5 1 1 5 1 1 1 5 4 0 1 3 1.1 -1 0 5 10 0 0 3 9 2 0 1 1	Case 1: 1 2 4 Case 2: -1

1155 – Power Transmission

DESA is taking a new project to transfer power. Power is generated by the newly established plant in Barisal. The main aim of this project is to transfer Power in Dhaka. As Dhaka is a megacity with almost 10 million people DESA wants to transfer maximum amount of power through the network. But as always occurs in case of power transmission it is tough to resist loss. So they want to use some regulators whose main aims are to divert power through several outlets without any loss.

Each such regulator has different capacity. It means if a regulator gets 100 units of power and its capacity is 80 units then remaining 20 units of power will be lost. Moreover each unidirectional link (connectors among regulators) has a certain capacity. A link with capacity 20 units cannot transfer power more than 20 units. Each regulator can distribute the input power among the outgoing links so that no link capacity is over flown. DESA wants to know the maximum amount of power which can be transmitted throughout the network so that no power loss occurs. That is the job you have to do.



(Do not try to mix the above description with the real power transmission.)

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

The input will start with a positive integer **N ($1 \leq N \leq 100$)** indicates the number of regulators. The next line contains **N** positive integers indicating the capacity of each regulator from **1** to **N**. All the given capacities will be positive and not greater than **1000**. The next line contains another positive integer **M** which is the number of links available among the regulators. Each of the following **M** lines contains three positive integers **i j C**. 'i' and 'j' are the regulator index ($1 \leq i, j \leq N, i \neq j, 1 \leq C \leq 1000$) and **C** is the capacity of the link. Power can be transferred from **ith** regulator to **jth** regulator. From a regulator **i** to another regulator **j**, there can be at most one link.

The next line contains two positive integers **B** and **D** ($1 \leq B, D$ and $B + D \leq N$). **B** is the number of regulators which are the entry point of the network. Power generated in Barisal must enter in the network through these entry points. Similarly **D** is the number of regulators connected to Dhaka. These links are special and have infinite capacity. Next line will contain **B+D** integers each of which is an index of regulator. The first **B** integers are the index of regulators connected with Barisal. Regulators connected with Barisal are not connected with Dhaka.

Output

For each case of input, print the case number and the maximum amount of power which can be transferred from Barisal to Dhaka.

Sample Input	Output for Sample Input
2 4 10 20 30 40 6 1 2 5 1 3 10 1 4 13 2 3 5 2 4 7 3 4 20 3 1 1 2 3 4 2 50 100 1 1 2 100 1 1 1 2	Case 1: 37 Case 2: 50

1156 - Jumping Frogs

With the increased use of pesticides, the local streams and rivers have become so contaminated that it has become almost impossible for the aquatic animals to survive.

Frog Fred is on the left bank of such a river. **N** rocks are arranged in a straight line from the left bank to the right bank. The distance between the left and the right bank is **D** meters. There are rocks of two sizes. The bigger ones can withstand any weight but the smaller ones start to drown as soon as any mass is placed on it. Fred has to go to the right bank where he has to collect a gift and return to the left bank where his home is situated.

He can land on every small rock at most one time, but can use the bigger ones as many times as he likes. He can never touch the polluted water as it is extremely contaminated.

Can you plan the itinerary so that the maximum distance of a single leap is minimized?

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers **N** ($0 \leq N \leq 100$) and **D** ($1 \leq D \leq 10^9$). The next line gives the description of the **N** stones. Each stone is defined by **S-M**. **S** indicates the type Big (**B**) or Small (**S**) and **M** ($0 < M < D$) determines the distance of that stone from the left bank. The stones will be given in increasing order of **M**.

Output

For each case of input, print the case number and the minimized maximum leap.

Sample Input	Output for Sample Input
3 1 10 B-5 1 10 S-5 2 10 B-3 S-6	Case 1: 5 Case 2: 10 Case 3: 7

1157 – LCS Revisited

LCS means 'Longest Common Subsequence' that means two non-empty strings are given; the longest subsequence that are common. Subsequence means removing 0 or more characters from a string.

Now you are given two non-empty strings **s** and **t**, your task is to find the number of distinct LCS of **s** and **t**. Since the result can be very big, print the result modulo **1000007**.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains two lines, the first line is the string **s** and the second line is the string **t**. You may assume that the strings are non-empty and consist only of lowercase letters and the length of each string is at most **1000**.

Output

For each case, print the case number and the number of distinct LCS of **s** and **t** modulo **1000007**.

Sample Input	Output for Sample Input
4 acbd acbd vnvn vn ab ba xyz abc	Case 1: 1 Case 2: 1 Case 3: 2 Case 4: 1

1158 - Anagram Division

Given a string s and a positive integer d you have to determine how many permutations of s are divisible by d .

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case contains a string s ($1 \leq s_{length} \leq 10$) and an integer d ($1 \leq d \leq 1001$). s will only contain decimal digits.

Output

For each case, print the case number and the number of permutations of s that are divisible by d .

Sample Input	Output for Sample Input
3 000 1 1234567890 1 123434 2	Case 1: 1 Case 2: 3628800 Case 3: 90

1159 - Batman

Batman is in deep trouble. You know that superheroes are there to help you when you are in trouble. But in Gotham city there is no trouble. So, 'no trouble' is actually the trouble for our Batman.

So, Batman is trying to solve ACM problems because he wants to be a good programmer like you :). But alas! He is not that smart. But still he is trying. He found 3 strings of characters. Now he wants to find the maximum string which is contained in all the three strings as a sub sequence. He wants to find the maximum length, not the sequence.

Now, Batman claims that he is a better programmer than you. So, you are solving the same problem. Can you solve faster? You are guaranteed that Batman will need 3 hours to solve the problem. So, you have to be faster than him.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case will contain a blank line and three strings in three lines. None of the string lengths will be greater than **50** and less than **1**. And the string will contain alphanumeric characters only.

Output

For each case, print one line containing the case number and the length of the largest subsequence.

Sample Input	Output for Sample Input
3 abcdef cdef dcdef aaaa bbbb ccca aaaa aaaa aaa	Case 1: 4 Case 2: 0 Case 3: 3

1160 – Discovering Paths in Grid

You are given a grid with 7 columns and N rows (numbered from 1 to N). 4 players will start their journey in that grid. Initially all of them are in the first row. A player can move diagonally to the next row. For example, a player is in row 3 and column 4, which can also be represented as (3, 4), this player has only two valid moves, (4, 3) and (4, 5). But if the player is in (3, 1), he has only one valid move, which is (4, 2). In their journey, any cell of the grid cannot be visited by more than one player.

In this problem you have to find, in how many ways all of the players can reach the N^{th} row.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case contains an integer N ($1 \leq N < 2^{31}$) in a line. The next line will contain 7 integers, which represent the state of first row. $r_i = 0$ represents that, there is no player in row i , otherwise r_i will represent the player number. You can assume that there will be exactly 4 players and are numbered from 1 to 4.

Output

For each case, print the case number and number of ways to reach the N^{th} row. You have to print the result modulo **1000000007**.

Sample Input	Output for Sample Input
3	Case 1: 1
1	Case 2: 0
1 0 2 0 3 0 4	Case 3: 3
2	
1 0 2 0 3 0 4	
2	
1 2 3 4 0 0 0	

1161 - Extreme GCD

All of you know that GCD means the greatest common divisor. So, you must have thought that this problem requires finding some sort of GCD. Don't worry, you are absolutely right!

Given **N** positive integers, not necessarily distinct, how many ways you can take **4** integers from the **N** numbers such that their GCD is **1**.

Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

Each case starts with an integer **N** ($4 \leq N \leq 10000$). The next line contains **N** integers separated by spaces. The integers will be positive and not greater than **10000**.

Output

For each case, print the case number and the number of ways you can take the integers as mentioned above.

Sample Input	Output for Sample Input
3 4 2 4 6 1 5 1 2 4 6 8 10 12 46 100 131 5 6 7 8 9 10	Case 1: 1 Case 2: 4 Case 3: 195

1162 – Min Max Roads

You live in a Big country where there are many bi-directional roads connecting the cities. Since the people of the country are quite intelligent, they designed the country such that there is exactly one path to go from one city to another. A path consists of one or more connected roads.

Here cities are denoted by integers and each road has a cost of traveling. Now you are given the information about the Country. And you are given some queries, each consists of two cities. You have to find the shortest and longest road in the path from one city to another.

Input

Input starts with an integer **T (≤ 5)**, denoting the number of test cases.

The first line of each case is a blank line. The next line contains **n ($2 \leq n \leq 10^5$)** denoting the number of cities. Then there will be **n-1** lines containing three integers each. They will be given in the form **u v w ($1 \leq u, v \leq n, 0 < w \leq 10^5, u \neq v$)** meaning that there is a road between **u** and **v** and the cost of the road is **w**.

The next line contains an integer **q ($1 \leq q \leq 25000$)** denoting the number of queries. Each of the next **q** lines contains two integers **x** and **y** ($1 \leq x, y \leq n, x \neq y$).

Output

For each case, print the case number in a single line. Then for each query **x y**, you should print one line containing the shortest and longest road along the path. See the samples for formatting.

Sample Input	Output for Sample Input
2 6 3 6 50 2 5 30 2 4 300 1 2 100 1 3 200 4 1 4 4 6 2 5 3 5 2 1 2 100 1 1 2	Case 1: 100 300 50 300 30 30 30 200 Case 2: 100 100

Note

Dataset is huge. Use faster i/o methods.

1163 - Bank Robbery

In one very cold morning, Mark decides to rob a bank. But while trying hacking into the security system, he found that it is locked by some random value. He also found a pattern on the random number, that is if he chops off the last digit of a number **A**, he gets a new number **B**. Then he calculates **(A-B)**. He checked the first few numbers of the security system which exactly equals **(A-B)**. Being very excited to have found the pattern, he learns that there are like 500 levels on the security system. He calculated all those numbers by hand but took a lot of time. As a sign of his accomplishment he left a note on the vault stating the pattern. You were the first officer on the crime scene and you've obtained the note. So if you can figure out **A** from **(A-B)**, you can rob the bank very quick!

By the way, Mark succeeded in robbing the bank but had a heart attack on the getaway car and crashed.

Input

Input starts with an integer **T (≤ 500)**, denoting the number of test cases.

Each line contains a single positive integer between **10** and **10^{18}** (inclusive), giving the value of **A-B**.

Output

For each case, print the case number and the possible values of **A** in ascending order. Separate consecutive numbers with a single space.

Sample Input	Output for Sample Input
4 31 18 12 17	Case 1: 34 Case 2: 19 20 Case 3: 13 Case 4: 18

1164 – Horrible Queries

World is getting more evil and it's getting tougher to get into the Evil League of Evil. Since the legendary Bad Horse has retired, now you have to correctly answer the evil questions of Dr. Horrible, who has a PhD in horribleness (but not in Computer Science). You are given an array of **n** elements, which are initially all **0**. After that you will be given **q** commands. They are -

1. **0 x y v** - you have to add **v** to all numbers in the range of **x** to **y** (inclusive), where **x** and **y** are two indexes of the array.
2. **1 x y** - output a line containing a single integer which is the sum of all the array elements between **x** and **y** (inclusive).

The array is indexed from **0** to **n - 1**.

Input

Input starts with an integer **T (≤ 5)**, denoting the number of test cases.

Each case contains two integers **n ($1 \leq n \leq 10^5$)** and **q ($1 \leq q \leq 50000$)**. Each of the next **q** lines contains a task in one of the following form:

0 x y v ($0 \leq x \leq y < n, 1 \leq v \leq 1000$)
1 x y ($0 \leq x \leq y < n$)

Output

For each case, print the case number first. Then for each query '**1 x y**', print the sum of all the array elements between **x** and **y**.

Sample Input	Output for Sample Input
2 10 5 0 0 9 10 1 1 6 0 3 7 2 0 4 5 1 1 5 5 20 3 0 10 12 1 1 11 12 1 19 19	Case 1: 60 13 Case 2: 2 0

Note

Dataset is huge. Use faster i/o methods.

1165 – Digit Dancing

Digits like to dance. One day, 1, 2, 3, 4, 5, 6, 7 and 8 stand in a line to have a wonderful party. Each time, a male digit can ask a female digit to dance with him, or a female digit can ask a male digit to dance with her, as long as their sum is a prime. Before every dance, exactly one digit goes to who he/she wants to dance with - either to its immediate left or immediate right.

For simplicity, we denote a male digit x by itself x , and denote a female digit x by $-x$. Suppose the digits are in order $\{1, 2, 4, 5, 6, -7, -3, 8\}$. If -3 wants to dance with 4 , she must go either to 4 's left, resulting $\{1, 2, -3, 4, 5, 6, -7, 8\}$ or his right, resulting $\{1, 2, 4, -3, 5, 6, -7, 8\}$.

Note that -3 cannot dance with 5 , since their sum $3+5 = 8$ is not a prime; 2 cannot dance with 5 , since they're both male.

Given the initial ordering of the digits, find the minimal number of dances needed for them to sort in increasing order (ignoring signs of course).

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case contains exactly eight integers in a single line. The absolute values of these integers form a permutation of $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

Output

For each test case, print the case number and the minimal number of dances needed. If they can never be sorted in increasing order, print **-1**.

Sample Input	Output for Sample Input
5	Case 1: 1
1 2 4 5 6 -7 -3 8	Case 2: 0
1 2 3 4 5 6 7 8	Case 3: 1
1 2 3 5 -4 6 7 8	Case 4: -1
1 2 3 5 4 6 7 8	Case 5: 3
2 -8 -4 5 6 7 3 -1	

1166 - Old Sorting

Given an array containing a permutation of **1** to **n**, you have to find the minimum number of swaps to sort the array in ascending order. A swap means, you can exchange any two elements of the array.

For example, let **n = 4**, and the array be **4 2 3 1**, then you can sort it in ascending order in just **1** swaps (by swapping **4** and **1**).

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains two lines, the first line contains an integer **n ($1 \leq n \leq 100$)**. The next line contains **n** integers separated by spaces. You may assume that the array will always contain a permutation of **1** to **n**.

Output

For each case, print the case number and the minimum number of swaps required to sort the array in ascending order.

Sample Input	Output for Sample Input
3 4 4 2 3 1 4 4 3 2 1 4 1 2 3 4	Case 1: 1 Case 2: 2 Case 3: 0

1167 - Dangerous Tunnels

Somewhere in the world, there are two tribes separated by mountains. The two tribes are named Kulolo and Gulolo, respectively, where Kulolo is at a higher altitude and Gulolo is at a lower altitude. Due to the limitation of geography, Gulolo has fewer resources than Kulolo. In order to transport resources from Kulolo to Gulolo efficiently, several tunnels were built inside the mountains between these two tribes. There are also some rest stations built for people to take a break during the transportation in the tunnels. More specifically, each terminal of a tunnel is one of Kulolo, Gulolo, or a rest station.

The structure of those tunnels is not stable. A dangerous degree has been estimated for each tunnel, due to its stability, in advance. A tunnel with a higher dangerous degree is considered to be more dangerous; that is, it is more probably to collapse. Kinglolo, the chief of Kulolo, would like to select some paths through the tunnels to Gulolo with little risk. In Kinglolo's opinion, the dangerous degree of a path is equal to the maximum dangerous degree of the tunnels in the path; and the dangerous degree of a set of paths is equal to the maximum dangerous degree of the paths in it. For example, consider Figure 1. The dangerous degrees of P_1 , P_2 and P_3 are, respectively 3, 5, and 6. And, the dangerous degree of $\{P_2, P_3\}$ is 6.

Since all tunnels are narrow, a limited quantity of resources can be transported along a path in one day. Therefore, every day, depending on the amount of resources needed to be transported, a different number, say k , of paths is required. Moreover, to avoid congestion, these k selected paths cannot pass any rest station in common. For example, in Figure 1, P_2 and P_3 can be selected at the same time; however, P_1 and P_2 cannot be selected at the same time, since they both pass r_2 .

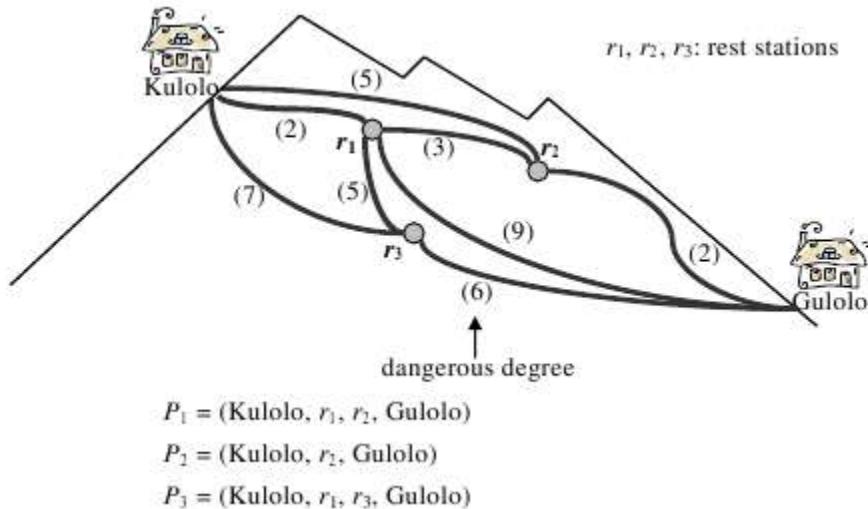


Figure 1: An example.

Kulolo has a higher altitude than all rest stations while Gulolo has a lower altitude than all rest stations. Kinglolo is a thoughtful chief. It is ensured that the altitudes of the rest stations on each

selected path are non-increasing, so that the path is more suitable for transportation. For example, in Figure 1, the path (**Kulolo**, r_3 , r_1 , **Gulolo**) will never be selected, since r_1 is higher than r_3 .

People in Kulolo believe that Kinglolo is the most brilliant man in the world, since he always selects a set of k paths that is as little dangerous as possible (i.e., the maximum dangerous degree of the selected paths is minimized). Now, given the data of the constructed tunnels, you are asked to find k paths that Kinglolo may select. In summary, the k selected paths, if exist, should satisfy the following:

1. all paths are from Kulolo to Gulolo,
2. no two paths pass the same rest station,
3. the altitudes of the rest stations on each path are non-increasing, and
4. the maximum dangerous degree of the paths is minimized.

For simplicity, only the maximum dangerous degree of the selected paths should be reported.

Input

Input starts with an integer T (≤ 30), denoting the number of test cases.

Each case starts with a blank line and a positive integer n ($0 < n \leq 100$) which indicates that there are n rest stations r_1, r_2, \dots, r_n . For ease of description, Kulolo and Gulolo are denoted by r_0 and r_{n+1} , respectively. We assume that r_i is higher than r_j for any ($0 \leq i < j \leq n+1$). The second line of each case contains a positive integer t (> 0) that specifies the number of tunnels. Each of the following t lines contains three integers p, q, d ($0 \leq p \leq n+1, 0 \leq q \leq n+1, p \neq q, 1 \leq d \leq 100000$) separated by white space, which indicate there is a tunnel with dangerous degree d connecting r_p and r_q . Then, a line containing a positive integer k ($1 \leq k \leq 10$) is provided, which is the number of paths that should be selected. You can assume that there is at most one tunnel between any two rest stations.

Output

For each case, print the case number and the maximum dangerous degree of the k paths that Kinglolo may select. If the solution does not exist, print '**no solution**'.

Sample Input	Output for Sample Input
<pre> 4 2 4 0 1 3 1 3 12 2 0 10 2 3 5 1 1 2 0 1 5 1 2 6 2 3 2 0 1 5 3 4 7 1 3 6 0 1 8 0 2 12 0 3 15 3 1 9 3 4 8 2 4 12 2 </pre>	<pre> Case 1: 10 Case 2: no solution Case 3: no solution Case 4: 12 </pre>

1168 - Wishing Snake

'Wishing Snake' is a computer game for children. In this game, there is a big board and a snake. The board contains 1000 check points numbered from 0 to 999. And the snake can go from any checkpoint to another without crossing other checkpoints. Initially the snake is at checkpoint 0.

Now **n** children come in front of the board and they start to wish. Each wish is like "I want to see the snake walking from checkpoint **u** to checkpoint **v**." And the snake can fulfill this wish by walking from checkpoint **u** to checkpoint **v** without crossing any other checkpoints. Each child can have multiple wishes. At first a child comes in front of the board and makes his wishes. After that a new child comes and makes his new wishes (that means not wished by any children yet). And it continues until the **n**th children.

After that the snake starts walking from one checkpoint to another. It can only walk from one checkpoint to another if any child had wished it. The snake wants to fulfill all the wishes done by all the children in one single path. Since you are the main designer of the game; you want to find whether it's possible or not.

Input

Input starts with an integer **T** (≤ 65), denoting the number of test cases.

Each case starts with an integer **n** ($1 \leq n \leq 100$). Then for each child an integer **k** ($k \geq 0$) is given. Each of the next **k** lines contains two integers **u v** ($0 \leq u, v < 1000, u \neq v$) denoting that the child wants to see the snake going from checkpoint **u** to checkpoint **v**. You may assume that all the wishes are distinct and correct. And the total number of wishes in any case is between **1** and **10000** (inclusive).

Output

For each case, print the case number and '**YES**' if it's possible, otherwise print '**NO**'.

Sample Input	Output for Sample Input
2 2 2 0 9 9 10 1 10 15 1 2 0 9 0 11	Case 1: YES Case 2: NO

1169 - Monkeys on Twin Tower

Being inspired by the ongoing popularity of animation films, the monkeys are trying to be smarter. They have realized that the only way to get smarter is to learn mathematics. Hence, they have started to do so. With the creative brains as they have, they are applying math in all aspects of life. Now, one of these mathematician monkeys are standing in front of a multi-storied twin tower. The twin tower is actually a couple of tall buildings standing parallel to each other. Each of the buildings has n floors. The ground floor is floor 0, the next one is floor 1 and so on. So, there are $2n$ floors in total in the twin tower. Each of these floors has a fruit inside it. The monkey knows in advance the amount of time required to eat the fruit in any floor. The monkey starts from the ground floor, climbs up toward the top of the buildings and has to eat exactly n fruits. From floor i , he has only two ways to go to floor $i + 1$. He can go the floor $i + 1$ of the same building that he is on in floor i . As he is a good jumper, it can be completed in no time. Also, he can go to floor $i + 1$ of the other building using a spiral stair connecting the two buildings. This will take a certain time. Please note that, the monkey can only move to floor $i + 1$ from floor i . He wants to figure out the minimum time required to eat n fruits. Can you verify how good his math is?

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each test case consists of five lines. The first line has a single integer n ($1 \leq n \leq 1000$), the number of floors in each building. The 2nd line contains n integers separated by a single space. These integers denote the number of seconds required to eat the fruit in each floor for the first building. The time is given in ascending order of the floor i.e. the first integer is the number of seconds required to eat the fruit in ground floor of the first building while the last integer is the time required for the fruit in the topmost floor. The next line, containing n integers, describes the same values for the right building. Each of the above $2n$ integers has a value between 1 and 100. The line four has $n - 1$ space separated integers. These values denote the time required to jump from the left building to the right one. So, the first integer is the number of seconds to jump from ground floor left building to 1st floor right building. Finally, the fifth line contains $n - 1$ more integers giving the time required for jumping from the right building to the left one. The jumping times have values between 1 and 50.

Output

For each case, print the case number and the minimum number of seconds required to eat n fruits.

Sample Input	Output for Sample Input
1 4 5 6 8 9 7 9 3 10 5 2 3 2 4 3	Case 1: 26

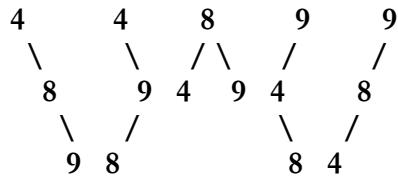
1170 - Counting Perfect BST

BST is the acronym for Binary Search Tree. A BST is a tree data structure with the following properties.

- i) Each BST contains a root node and the root may have zero, one or two children. Each of the children themselves forms the root of another BST. The two children are classically referred to as left child and right child.
- ii) The left subtree, whose root is the left children of a root, contains all elements with key values less than or equal to that of the root.
- iii) The right subtree, whose root is the right children of a root, contains all elements with key values greater than that of the root.

An integer m is said to be a perfect power if there exists integer $x > 1$ and $y > 1$ such that $m = x^y$. First few perfect powers are $\{4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 81, 100, 121, 125, 128, 144, \dots\}$. Now given two integer a and b we want to construct BST using all perfect powers between a and b , where each perfect power will form the key value of a node.

Now, we can construct several BSTs out of the perfect powers. For example, given $a = 1$ and $b = 10$, perfect powers between a and b are 4, 8, 9. Using these we can form the following five BSTs.



In this problem, given a and b , you will have to determine the total number of BSTs that can be formed using perfect powers between a and b .

Input

Input starts with an integer T (≤ 20000), denoting the number of test cases.

Each case of input contains two integers: a and b ($1 \leq a \leq b \leq 10^{10}$, $b - a \leq 10^6$) as defined in the problem statement.

Output

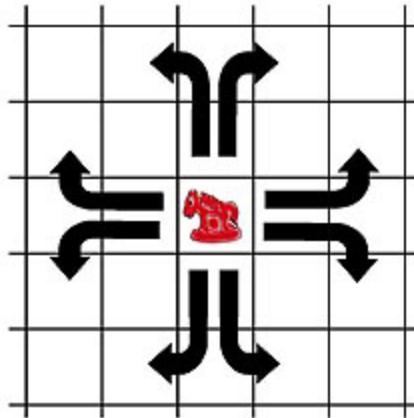
For each case, print the case number and the total number of distinct BSTs that can be formed by the perfect powers between a and b . Output the result modulo 100000007 .

Sample Input	Output for Sample Input
4 1 4 5 10 1 10 1 3	Case 1: 1 Case 2: 2 Case 3: 5 Case 4: 0

1171 - Knights in Chessboard (II)

Given an $m \times n$ chessboard where some of the cells are broken. Now you are about to place chess knights in the chessboard. You have to find the maximum number of knights that can be placed in the chessboard such that no two knights attack each other. You can't place knights in the broken cells.

Those who are not familiar with chess knights, note that a chess knight can attack eight positions in the board as shown in the picture below.



Input

Input starts with an integer T (≤ 125), denoting the number of test cases.

Each case starts with a blank line. The next line contains three integers m , n , K ($1 \leq m, n \leq 200$). Here m and n corresponds to the number of rows and the number of columns of the board respectively. Each of the next K lines will contain two integers x , y ($1 \leq x \leq m$, $1 \leq y \leq n$) denoting that the cell (x, y) is broken already. No broken cell will be reported more than once.

Output

For each case of input, print the case number and the maximum number of knights that can be placed in the board considering the above restrictions.

Sample Input	Output for Sample Input
2 8 8 0 2 5 4 1 3 1 4 2 3 2 4	Case 1: 32 Case 2: 6

1172 – Krypton Number System

All of you must have heard the name of the planet Krypton. If you can't remember the planet, don't worry. Planet Krypton is the origin of Superman that means the mother planet where Superman was born. Superman was sent to earth by his parents when the planet was about to explode. Legends say that only few people of the planet survived from that explosion.

The inhabitants of the planet are called 'Kryptonians'. Kryptonians, though otherwise completely human, were superior both intellectually and physically to natives of Earth. One of the most common differences is their number system. The number system is denoted below:

- 1) The base of the number system is unknown, but legends say that the base lies between 2 and 6.
- 2) Kryptonians don't use a number where two adjacent digits are same. They simply ignore these numbers. So, 112 is not a valid number in Krypton.
- 3) Numbers should not contain leading zeroes. So, 012 is not a valid number.
- 4) For each number, there is a score. The score can be found by summing up the squares of differences of adjacent numbers. For example 1241 has the score of
- 5) $(1-2)^2 + (2-4)^2 + (4-1)^2 = 1 + 4 + 9 = 14$.
- 6) All the numbers they use are integers.

Now you are planning to research on their number system. So, you assume a base and a score. You have to find, how many numbers can make the score in that base.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case contains two integers denoting **base ($2 \leq \text{base} \leq 6$)** and **score ($1 \leq \text{score} \leq 10^9$)**. Both the integers will be given in decimal base.

Output

For each case print the case number and the result modulo 2^{32} . Check the samples for details. Both the case number and result should be reported in decimal base.

Sample Input	Output for Sample Input
2 6 1 5 5	Case 1: 9 Case 2: 80

1173 – The Vindictive Coach

The coach of a football team, after suffering for years the adverse comments of the media about his tactics, decides to take his revenge by presenting his players in a line-up in such a way that the TV cameras would be compelled to zigzag in a ridiculous bobbing motion, by alternating taller and shorter players. However, the team captain objects that he must be the first of the line by protocolary reasons, and that he wants to be seen in the best possible light: that is, he should not have a taller colleague next to him unless there is no alternative (everyone else is taller than him). Even in this case, the height difference should be as small as possible, while maintaining the zigzag arrangement of the line.

With this condition the coach addresses an expert in computation (i.e. you) to help him find the number of different alignments he may make, knowing that all players have a different height. They are always numbered by stature starting by **1** as the shortest one. Of course the number of players may be arbitrary, provided it does not exceed **50**.

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case contains two positive integers **N** ($1 \leq N \leq 50$) and **m** ($1 \leq m \leq N$) separated by a blank space. **N** represents the number of players in the line-up and **m** is the captain's number, who as told is always the first of the line.

Output

For every case, print the case number and the number of possible alignments under the above conditions. Output the result modulo 2^{64} .

Sample Input	Output for Sample Input
4 3 1 3 3 4 1 7 2	Case 1: 1 Case 2: 1 Case 3: 1 Case 4: 16

1174 – Commandos

A group of commandos were assigned a critical task. They are to destroy an enemy head quarter. The enemy head quarter consists of several buildings and the buildings are connected by roads. The commandos must visit each building and place a bomb at the base of each building. They start their mission at the base of a particular building and from there they disseminate to reach each building. The commandos must use the available roads to travel between buildings. Any of them can visit one building after another, but they must all gather at a common place when their task is done. In this problem, you will be given the description of different enemy headquarters. Your job is to determine the minimum time needed to complete the mission. Each commando takes exactly one unit of time to move between buildings. You may assume that the time required to place a bomb is negligible. Each commando can carry unlimited number of bombs and there is an unlimited supply of commando troops for the mission.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

The first line of each case starts with a positive integer **N ($1 \leq N \leq 100$)**, where **N** denotes the number of buildings in the head quarter. The next line contains a positive integer **R**, where **R** is the number of roads connecting two buildings. Each of the next **R** lines contain two distinct numbers **u v ($0 \leq u, v < N$)**, this means there is a road connecting building **u** to building **v**. The buildings are numbered from **0** to **N-1**. The last line of each case contains two integers **s d ($0 \leq s, d < N$)**. Where **s** denotes the building from where the mission starts and **d** denotes the building where they must meet. You may assume that two buildings will be directly connected by at most one road. The input will be given such that, it will be possible to go from any building to another by using one or more roads.

Output

For each case, print the case number and the minimum time required to complete the mission.

Sample Input	Output for Sample Input
2 4 3 0 1 2 1 1 3 0 3 2 1 0 1 1 0	Case 1: 4 Case 2: 1

1175 – Jane and the Frost Giants

Jane is one of the most talented young programmers as well as an astrophysicist. Recently she discovered a planet and named it Jotunheim - the world of giants. As you already guessed that the inhabitants are all giants. Among them the Frost Giants are the most evil ones. Before Jane could publicly announce her great discovery, the Frost Giants came and captured her in a maze. Since the Giants would be discovered to the universe because of her, that's why they lit fires on some positions in the maze to kill her.

You are given Jane's location in the maze and the positions of the fires lit by the Frost Giants whom are always keeping an eye on her; you must find out whether Jane can escape from the maze before fire catches her, and how fast she can do it.

The Maze is defined as a 2D grid and the locations are defined as squares. The cost of each move is one square per minute. In each move, Jane can move vertically or horizontally but not diagonally. She cannot move to a square which is blocked by an obstacle, or which is already burning. If a square has fire in it, in the next minute, fires spread to its adjacent **non-obstacle** squares (vertically or horizontally). Jane can escape from the maze from any squares that borders the edge of the maze.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

The first line of each test case contains the two integers **R** and **C**, separated by spaces, with $1 \leq R, C \leq 200$. The following **R** lines of the test case each contain one row of the maze. Each of these lines contains exactly **C** characters, and each of these characters is one of:

1. # an obstacle
2. . a free location
3. J Jane's initial position in the maze (there will be exactly one 'J' in the maze)
4. F Position of a Fire

Output

For each case, print the case number and '**IMPOSSIBLE**' if Jane cannot escape from the maze before fire reaches her, or the earliest time for Jane to safely escape from the maze, in minutes.

Sample Input	Output for Sample Input
2 4 5 ##.## #JF.# #...# #...# 3 3 ### #J. .F	Case 1: 3 Case 2: IMPOSSIBLE

1176 - Getting a T-shirt

In programming contests, a common problem for the contestants is to get a suited T-shirt. Sometimes people gets too long or too short T-shirts. So, this time I have planned to ask the authority to manage the T-shirts such that everyone gets a suitable one. From my past experience, it's known that there are 6 available sizes of T-shirts and they are XXL, XL, L, M, S, and XS. And exactly two sizes of the T-shirts suit a person.

Now, for a contest there are T-shirts of **N** colors and **M** contestants. And for each color, all the 6 sizes are available. So, there are **6*N** T-shirts. And you are given the suitable sizes for each contestant. You have to distribute the T-shirts to the contestants such that everyone gets a suitable size. Only size matters, color is not an issue. Now you have to decide whether it's possible or not.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

The first line of each test case contains two integers **N** and **M**, separated by spaces, with $1 \leq N, M \leq 50$. Each of the next **M** lines will contain two sizes as described earlier.

Output

For each case, print the case number and "YES" or "NO" depending on whether it's possible to distribute the T-shirts or not.

Sample Input	Output for Sample Input
3 3 6 L XL XL L XXL XL S XS M S M L 1 4 S XL L S L XL L XL 1 1 L M	Case 1: YES Case 2: NO Case 3: YES

1177 - Angry Programmer

You, a programmer of an important software house, have been fired because you didn't solve an important problem that was assigned to you. You are very furious and want to take revenge on your boss, breaking the communication between his computer and the central server.

The computer of your boss and the central server are in the same network, which is composed of many machines (computers) and wires linking pairs of those machines. There is at most one wire between any pair of machines and there can be pairs of machines without a wire between them.

To accomplish your objective, you can destroy machines and wires, but you can't destroy neither the computer of your boss, nor the central server, because those machines are monitored by security cameras. You have estimated the cost of blowing up each machine and the cost of cutting each wire in the network.

You want to determine the minimum cost of interrupting the communication between your boss' computer and the central server. Two computers **A** and **B** can communicate if there is a sequence of undestroyed machines x_1, x_2, \dots, x_n such that $x_1 = A$, $x_n = B$ and x_i is linked with x_{i+1} with an uncut wire (for each $1 \leq i < n$).

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

Each case starts with two integers **M** ($2 \leq M \leq 50$) and **W** ($0 \leq W \leq 1200$) where **M** denotes the number of machines and **W** denotes the number of wires. The ids of the boss' machine and the server are **1** and **M** respectively. The next line contains **M - 2** integers denoting the cost for destroying the machines from **2** to **M - 1** respectively. Each of the next **W** lines contains three integers **i j c**, meaning that the wire between machine **i** and **j** can be destroyed with the cost of **c**. All the destroying costs are between **0** and **10^5** .

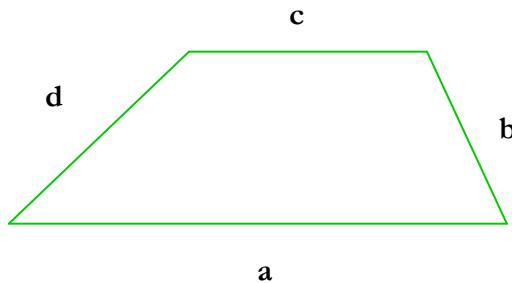
Output

For each case, print the case number and the minimum cost of interrupting the communication between the computer of your boss and the central server.

Sample Input	Output for Sample Input
2 4 4 2 5 1 2 3 1 3 3 2 4 1 3 4 3 4 4 2 2 1 2 3 1 3 3 2 4 1 3 4 3	Case 1: 4 Case 2: 3

1178 - Trapezium

You are given the length of the four sides of a trapezium; you have to calculate the area. In geometry a 4-sided figure with exactly one pair of parallel sides is called a trapezium.



Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

Each case contains four real numbers **a** **b** **c** **d** denoting the sides of the trapezium. Here **a** and **c** denote the parallel sides. You can safely assume that the given trapezium is valid. Each of the numbers will be positive and not more than **200**. And no number contains more than 4 digits after the decimal point.

Output

For each case, print the case number and the area. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 6 5 12 5.0 9 5 6 4	Case 1: 36 Case 2: 30.000000

1179 – Josephus Problem

The historian Flavius Josephus relates how, in the Romano-Jewish conflict of 67 A.D., the Romans took the town of Jotapata which he was commanding. Escaping, Josephus found himself trapped in a cave with 40 companions. The Romans discovered his whereabouts and invited him to surrender, but his companions refused to allow him to do so. He therefore suggested that they kill each other, one by one, the order to be decided by lot. Tradition has it that the means for affecting the lot was to stand in a circle, and, beginning at some point, count round, every third person being killed in turn. The sole survivor of this process was Josephus, who then surrendered to the Romans. Which begs the question: had Josephus previously practiced quietly with 41 stones in a dark corner, or had he calculated mathematically that he should adopt the 31st position in order to survive?

Now you are in a similar situation. There are **n** persons standing in a circle. The persons are numbered from **1** to **n** circularly. For example, **1** and **n** are adjacent and **1** and **2** are also. The count starts from the first person. Each time you count up to **k** and the **kth** person is killed and removed from the circle. Then the count starts from the next person. Finally one person remains. Given **n** and **k** you have to find the position of the last person who remains alive.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains two positive integers **n** ($1 \leq n \leq 10^5$) and **k** ($1 \leq k < 2^{31}$).

Output

For each case, print the case number and the position of the last remaining person.

Sample Input	Output for Sample Input
6	Case 1: 2
2 1	Case 2: 1
2 2	Case 3: 3
3 1	Case 4: 3
3 2	Case 5: 2
3 3	Case 6: 3
4 6	

1180 – Software Company

A software developing company has been assigned two programming projects. As both projects are within the same contract, both must be handed in at the same time. It does not help if one is finished earlier.

This company has **n** employees to do the jobs. To manage the two projects more easily, each is divided into **m** independent subprojects. Only one employee can work on a single subproject at one time, but it is possible for two employees to work on different subprojects of the same project simultaneously. Our goal is to finish the projects as soon as possible.

Input

Input starts with an integer **T (≤ 12)**, denoting the number of test cases.

Each case starts with two integers **n ($1 \leq n \leq 100$)**, and **m ($1 \leq m \leq 100$)**. Each of the next **n** lines contains two integers which specify how much time in seconds it will take for the specified employee to complete one subproject of each project. So if the line contains **x** and **y**, it means that it takes the employee **x** seconds to complete a subproject from the first project, and **y** seconds to complete a subproject from the second project.

Output

For each case, print the case number and the minimum amount of time in seconds after which both projects can be completed. The input will be such that answer will be within **50000**.

Sample Input	Output for Sample Input
1 3 20 1 1 2 4 1 6	Case 1: 18

1181 - Odd Looking Average

We are all quite familiar with finding 'average'. Let us define new kind of average of a function.

Given a function $f(x)$ and two values a and b ($a \leq b$) if we take all the numbers (not necessarily integers) from a to b then

$$y = \frac{\text{Summation of } f(x)(a \leq x \leq b)}{\text{Total numbers in } [a, b]}$$

Now for $f(x) = x^k$ you are given k , a and b , you have to find the average according to the description.

Input

Input starts with an integer T (≤ 400), denoting the number of test cases.

Each case contains an integer k ($1 \leq k \leq 4$) and two real numbers a and b ($0 < a \leq b \leq 10$).

Output

For each case, print the case number and the average. Error less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 1 2 2 2 2 2.0 1 2.0 4 2 2 4.0	Case 1: 2 Case 2: 4 Case 3: 3 Case 4: 9.3333333333

1182 - Parity

Given an integer **n**, first we represent it in binary. Then we count the number of ones. We say **n** has odd parity if the number of one's is odd. Otherwise we say **n** has even parity. **21 = (10101)₂** has odd parity since the number of one's is **3**. **6 = (110)₂** has even parity.

Now you are given **n**, we have to say whether **n** has even or odd parity.

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case contains an integer **n** ($1 \leq n < 2^{31}$).

Output

For each case, print the case number and '**odd**' if **n** has odd parity, otherwise print '**even**'.

Sample Input	Output for Sample Input
2 21 6	Case 1: odd Case 2: even

1183 - Computing Fast Average

Given an array of integers (**0** indexed), you have to perform two types of queries in the array.

1. **1 i j v** - change the value of the elements from i^{th} index to j^{th} index to **v**.
2. **2 i j** - find the average value of the integers from i^{th} index to j^{th} index.

You can assume that initially all the values in the array are **0**.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

Each case contains two integers: **n** ($1 \leq n \leq 10^5$), **q** ($1 \leq q \leq 50000$), where **n** denotes the size of the array. Each of the next **q** lines will contain a query of the form:

1 i j v ($0 \leq i \leq j < n$, $0 \leq v \leq 10000$)
2 i j ($0 \leq i \leq j < n$)

Output

For each case, print the case number first. Then for each query of the form '**2 i j**' print the average value of the integers from **i** to **j**. If the result is an integer, print it. Otherwise print the result in '**x/y**' form, where **x** denotes the numerator and **y** denotes the denominator of the result and **x** and **y** are relatively prime.

Sample Input	Output for Sample Input
1 10 6 1 0 6 6 2 0 1 1 1 1 2 2 0 5 1 0 3 7 2 0 1	Case 1: 6 16/3 7

Note

Dataset is huge. Use faster i/o methods.

1184 – Marriage Media

You run a marriage media. You take some profiles for men and women, and your task is to arrange as much marriages as you can. But after reading their bio-data you have found the following criteria.

1. No man will marry a woman if their height gap is greater than **12** inches.
2. No woman will marry a man if their age gap is greater than **5** years.
3. A couple can be formed if either both are not divorced or both are divorced.
4. Of course, a man can marry a single woman and vice versa.

Now you are given the bio-data of some men and women, you have to arrange the maximum number of marriages considering the given criteria.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case contains two integer **m, n ($1 \leq m, n \leq 50$)**. Each of the next **m** lines will contain the information for a man, and each of the next **n** lines will contain the information for a woman. An information will contain three integers denoting the height in inches, age in years and **1** or **0** depending on they are divorced or not respectively. Assume that Height will be between **50** and **80**, age will be between **20** and **50**.

Output

For each case, print the case number and the maximum number of marriages you can arrange.

Sample Input	Output for Sample Input
2 2 2 70 30 0 60 20 0 71 25 0 71 35 0 1 1 70 30 1 70 30 0	Case 1: 2 Case 2: 0

1185 - Escape

Hanzo Hattori rescued Princess Nakururu and now they are planning to escape from the castle, because it's going to explode. Now the problem is that Nakururu is not quite well that's why she can't run all the way. And Hattori is also injured. That's why he can't carry her all the way.

So, they made a plan. From their current position, Hanzo will carry Nakururu to the next place. Then they both will run to the next place. Then again Hanzo will carry Nakururu to the next place. And they will continue this procedure to go from one place to another.

Each place can be treated as a node. Nodes are connected by bi-directional roads. All the nodes are numbered as integers. Initially they are at node **1**. And both of them will run to the next node.

Now given the description of the nodes and roads you have to count the number of possible nodes which can be entered by Hanzo carrying Nakururu with him. Initially Hattori doesn't carry Nakururu.

Input

Input starts with an integer **T (≤ 210)**, denoting the number of test cases.

Each case starts with a blank line. The next line will contain two integers: **n ($1 \leq n \leq 100$)** and **m ($0 \leq m \leq n*(n-1)/2$)**, indicating the number of nodes and the number of roads respectively. The next **m** lines, each will contain two integers **a b ($1 \leq a, b \leq n, a \neq b$)** indicating that there is a road between node **a** and **b**. All the given roads will be distinct.

Output

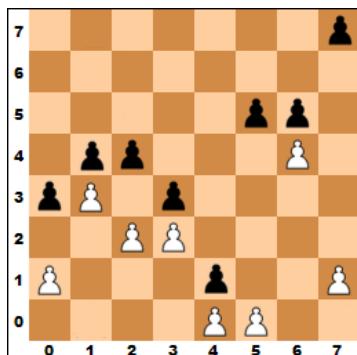
For each case, print the case number and number of nodes that can be entered by Hanzo, carrying Nakururu with him.

Sample Input	Output for Sample Input
2 4 3 1 2 2 3 3 4 5 4 1 2 2 3 1 3 3 4	Case 1: 2 Case 2: 4

1186 – Incredible Chess

You are given an $n \times n$ chess board. Only pawn is used in the 'Incredible Chess' and they can move forward or backward. In each column there are two pawns, one white and one black. White pawns are placed in the lower part of the board and the black pawns are placed in the upper part of the board.

The game is played by two players. Initially a board configuration is given. One player uses white pieces while the other uses black. In each move, a player can move a pawn of his piece, which can go forward or backward any positive integer steps, but it cannot jump over any piece. White gives the first move. The game ends when there is no move for a player and he will lose the game. Now you are given the initial configuration of the board. You have to write a program to determine who will be the winner.



Example of a Board

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with an integer n ($3 \leq n \leq 200$) denoting the dimension of the board. The next line will contain n integers, W_0, W_1, \dots, W_{n-1} giving the position of the white pieces. The next line will also contain n integers, B_0, B_1, \dots, B_{n-1} giving the position of the black pieces. W_i means the row position of the white piece of i^{th} column. And B_i means the row position of the black piece of i^{th} column. You can assume that $(0 \leq W_i < B_i < n)$ for $(0 \leq i < n)$ and at least one move is remaining.

Output

For each case, print the case number and 'white wins' or 'black wins' depending on the result.

Sample Input	Output for Sample Input
2 6 1 3 2 2 0 1 5 5 5 3 1 2 7 1 3 2 2 0 4 0 3 4 4 3 1 5 6	Case 1: black wins Case 2: white wins

1187 – Lining up Students

Today is the Independence Day in Bangladesh. So, n students from various schools are asked to join and line up in a large hall and they were given distinct ids from 1 to n . The ids were distributed from shortest to tallest; so, the shortest student got id 1 and the tallest student got id n . Every student's height was measured by a machine which can calculate up to micrometers. And so, every student's height was distinct!

When the teachers asked the students to form a line, the students did form a line from left to right but didn't follow any patterns, so the line looks messy rather than organized. Now, the teachers decided to make the line from shortest to tallest. So, they asked them to line up from id 1 to n . But one of the teachers asked each of the students to count the number of taller students who are currently in left. As the students are quite clever, they did this quickly and informed that teacher. The teacher noted this information in a notebook and then asked them to line up from 1 to n . And finally the desired line was formed.

And finally the day was over with a nice celebration. However, that teacher was from CSE department and the next day he went to his class and showed the notebook to the students and asked them to find the initial order formed by them. There were only 12/13 students and almost all raised their hands. But the teacher said, '**I am asking to solve this problem if $n = 10^5$** '. The students looked each other's faces and storming their heads to solve this problem. As you are one of the talented students of that great teacher, can you solve this problem?

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case contains two lines. In first line there will be an integer n ($1 \leq n \leq 10^5$) denoting the number of students. The next line contains n space separated integers. The i^{th} integer of that line denotes the number of students who are in left and taller than the i^{th} student. Assume that the given data is valid.

Output

For each case, print the case number and the id of the leftmost student.

Sample Input	Output for Sample Input
3 3 0 0 0 3 0 1 2 3 0 1 0	Case 1: 1 Case 2: 3 Case 3: 2

Note

Dataset is huge. Use faster I/O methods.

1188 – Fast Queries

Given an array of N integers indexed from 1 to N , and q queries, each in the form i j , you have to find the number of distinct integers from index i to j (inclusive).

Input

Input starts with an integer T (≤ 5), denoting the number of test cases.

The first line of a case is a blank line. The next line contains two integers N ($1 \leq N \leq 10^5$), q ($1 \leq q \leq 50000$). The next line contains N space separated integers forming the array. The integers range in $[0, 10^5]$.

Each of the next q lines will contain a query which is in the form i j ($1 \leq i \leq j \leq N$).

Output

For each test case, print the case number in a single line. Then for each query you have to print a line containing number of distinct integers from index i to j .

Sample Input	Output for Sample Input
1 8 5 1 1 1 2 3 5 1 2 1 8 2 3 3 6 4 5 4 8	Case 1: 4 1 4 2 4

Note

Dataset is huge. Use faster I/O methods.

1189 – Sum of Factorials

Given an integer n , you have to find whether it can be expressed as summation of factorials. For given n , you have to report a solution such that

$$n = x_1! + x_2! + \dots + x_n! \quad (x_i < x_j \text{ for all } i < j)$$

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 10^{18}$).

Output

For each case, print the case number and the solution in summation of factorial form. If there is no solution then print '**impossible**'. There can be multiple solutions, any valid one will do. See the samples for exact formatting.

Sample Input	Output for Sample Input
4	Case 1: 1!+3!
7	Case 2: 0!+3!
7	Case 3: 1!+2!+3!
9	Case 4: impossible
11	

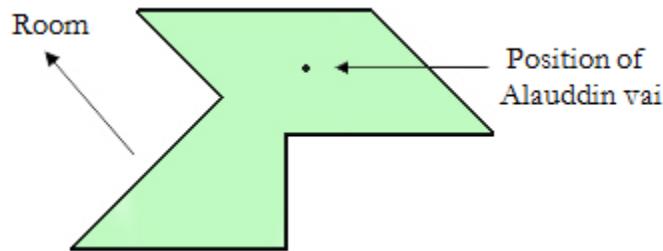
Note

Be careful about the output format; you may get wrong answer for wrong output format.

1190 – Sleepwalking

All of you know about "Alauddin vai". But what you don't know is, he sometimes walks while sleeping. Yes, some kind of sleep walking.

Now one day he was sleeping in a strange room. But after waking up he doubts that whether he is inside the room or not.



You may assume that the room can be modeled as a polygon whose coordinates are 2D integer points. Alauddin vai can be thought as a 2D integer point. Now you are given the configuration of the room and the position of Alauddin vai (after waking up). You have to decide whether he is still in the room or not. Consider him inside if he is on the boundary of the polygon.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with an integer **n** ($3 \leq n \leq 100$) denoting the number of vertices of the room. The next line contains $2n$ integers, where the i^{th} pair $x_i \ y_i$ denote the co-ordinate of the i^{th} vertex of the room. The vertices will be given in anticlockwise order. The next line contains an integer **q** ($1 \leq q \leq 300$) denoting the number of queries. Each of the next **q** lines contains two integers **x** **y** denoting the co-ordinate of Alauddin vai. The absolute value of the given co-ordinates will be less than **10001**.

Output

For each case, print the case number in a single line. Then print **q** lines, containing the answer for the queries as given in input. Print '**Yes**' if he is inside the room, print '**No**' otherwise.

Sample Input	Output for Sample Input
2 3 0 0 10 0 0 20 2 5 5 10 10 3 0 0 3 3 0 3 1 5 5	Case 1: Yes No Case 2: No

1191 – Bar Codes

A bar-code symbol consists of alternating dark and light bars, starting with a dark bar on the left. Each bar is a number of units wide. Figure 1 shows a bar-code symbol consisting of 4 bars that extend over $1+2+3+1=7$ units.

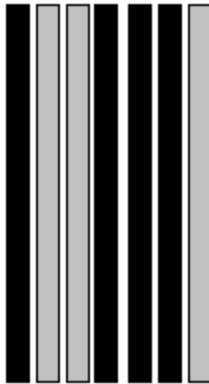


Figure 1: Bar-code over 7 units with 4 bars

In general, the bar code **BC(n, k, m)** is the set of all symbols with **k** bars that together extend over exactly **n** units, each bar being at most **m** units wide. For instance, the symbol in Figure 1 belongs to BC(7,4,3) but not to BC(7,4,2). Figure 2 shows all 16 symbols in BC(7,4,3). Each '1' represents a dark unit, each '0' represents a light unit.

0: 1000100 4: 1001110 8: 1100100 12: 1101110
1: 1000110 5: 1011000 9: 1100110 13: 1110010
2: 1001000 6: 1011100 10: 1101000 14: 1110100
3: 1001100 7: 1100010 11: 1101100 15: 1110110

Figure 2: All symbols of BC(7,4,3)

Input

Input starts with an integer **T (≤ 20000)**, denoting the number of test cases.

Each case contains three integers: **n, k, m ($1 \leq k, m \leq n \leq 50$)**.

Output

For each case, print the case number and **BC(n, k, m)**.

Sample Input	Output for Sample Input
2 7 4 3 7 4 2	Case 1: 16 Case 2: 4

1192 - Left Right

Two players, Alice and Bob are playing a strange game in a $1 \times n$ board. The cells are numbered from **0** to **n-1**, where the left most cell is marked as cell **0**. Each cell can contain at most one piece.



Fig 1: an example

There are two kinds of pieces, gray and white. Alice moves all gray pieces, and bob moves all white ones. The pieces alternate, that is, leftmost piece is gray, next is white, next to that is gray, then it's white again, and so on. There will always be equal number of black and gray pieces. Alice can only move pieces to the right. Bob can only move pieces to the left.

At each move, a player selects one piece and moves that piece, either to its left (Bob) or to its right (Alice), any number of cells (at least 1) but, it can neither jump over other pieces, nor it can move outside the board. The players alternate their turns.

For example, if Alice decides to move the left most gray piece, these two moves are available to her.



Fig 2: Moving the gray piece one cell to the right



Fig 3: Moving the gray piece two cells to the right

Alice moves first. The game ends, when someone is unable to make any move, and loses the game. You can assume that, both of them play optimally (that is, if it is possible to apply a strategy that will ensure someone's win, he/she will always use that strategy).

Now you are given a configuration of a board, you have to find the winner.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with a line containing an integer **k** ($1 \leq k \leq 100$) denoting the number of gray pieces in the board. The next line contains $2k$ distinct integers (in ascending order) denoting the position of the pieces. The first integer denotes a gray piece, the second integer denotes a white piece, the next integer denotes a gray piece and so on. All the integers will lie in the range $[0, 10^9]$. Assume that **n** is sufficiently large to contain all the pieces. And at least one move is remaining.

Output

For each case, print the case number and '**Alice**' or '**Bob**' depending on the winner of the game.

Sample Input	Output for Sample Input
<pre>2 2 0 3 7 9 2 1 3 7 9</pre>	<pre>Case 1: Alice Case 2: Bob</pre>

1193 - Dice (II)

You have **N** dices; each of them has **K** faces numbered from 1 to **K**. Now you can arrange the **N** dices in a line. If the summation of the top faces of the dices is **S**, you calculate the score as the multiplication of all the top faces.

Now you are given **N**, **K**, **S**; you have to calculate the summation of all the scores.

Input

Input starts with an integer **T** (≤ 25), denoting the number of test cases.

Each case contains three integers: **N** ($1 \leq N \leq 1000$) **K** ($1 \leq K \leq 1000$) **S** ($0 \leq S \leq 15000$).

Output

For each case print the case number and the result modulo **100000007**.

Sample Input	Output for Sample Input
5 1 6 3 2 9 8 500 6 1000 800 800 10000 2 100 10	Case 1: 3 Case 2: 84 Case 3: 74335590 Case 4: 33274428 Case 5: 165

1194 - Colored T-Shirts

N students are standing in a line. They are wearing colored t-shirts. For simplicity, let's assume that the color of each person is an integer between **1** and **16**. Like color **1** represents red, color **2** represents blue, and so on.

Now the students with same colored t-shirts want to join together. So, they are thinking what to do. That's why you are here to the rescue! You have to solve this problem. You have to say the minimum number of swaps necessary to reformat the line such that the people with same colored t-shirt come together. A swap means changing position between two adjacent people in the line.

Input

Input starts with an integer **T** (≤ 15), denoting the number of test cases.

Each case starts with two integers **N** ($1 \leq N \leq 10^5$) and **m** ($1 \leq m \leq 16$), where **m** denotes the number of total colors. So, the students in the line will have a t-shirt color between **1** and **m**. The next line contains **N** space separated integers. Each of these integers will lie between **1** and **m**, denoting the color of the i^{th} person in the line.

Output

For each case, print the case number and the minimum number of swaps necessary to arrange them according to the description.

Sample Input	Output for Sample Input
3 4 2 1 2 1 2 6 4 2 1 4 3 1 2 8 6 1 3 2 5 5 4 5 2	Case 1: 1 Case 2: 6 Case 3: 5

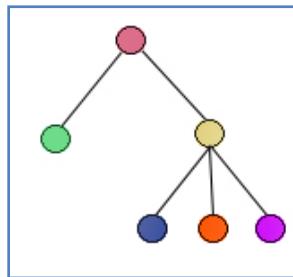
Note

Dataset is huge. Use faster i/o methods.

1195 – Similar Trees

Given two trees, you have to find whether they are same or not. A tree is a connected graph with no cycles.

In this problem a tree is described as a sequence of zeroes and ones. Initially you are on the root of the tree. If you get a **1**, you declare a new node and you go to that node. The parent node of this node is the node you came from. If you get a **0**, you will return back to its parent node. The trees will be given such that after traversing all the nodes, you will return back to the root.



The given tree can be described in two ways.

1. 1011010100
2. 1101010010

In this problem you are given two trees. You have to determine whether they are same or not. Two trees will be same if we start from their roots, we will get same shaped trees.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case contains two lines. In each line, a string will be given as a tree configuration. You can safely assume that the trees are valid and the length of any string will be between **0** and **10000** (inclusive). You can assume that the length of both the trees will be same.

Output

For each case print the case number and '**Same**' or '**Different**' depending on whether the trees are same or not.

Sample Input	Output for Sample Input
3 1011010100 1101010010	Case 1: Same Case 2: Same Case 3: Different
1101010010 1011110000	

1196 - Inhabitants

The kingdom of 'Geometry Land' is huge and the number of people living here is also huge. One day, The King wanted to find how many people living in this kingdom. So, he asked all the inhabitants of the kingdom to stay in the kingdom for one day.

The kingdom can be modeled in a 2D plane as a convex polygon. The King sent many messengers to find all the people they can find. The messengers were not good at geometry, so, they noted the positions of the people as 2D co-ordinates using GPS.

Now you are given the map and the positions of the people of the kingdom. You have to find whether a person belongs to the kingdom or not. You can safely assume that the person is an inhabitant of the kingdom if and only if his position is inside the kingdom. If a person lies in the boundary of the map, you can consider him inside.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a blank line and an integer **n ($3 \leq n \leq 10^5$)**, denoting the number of vertices of the convex polygon that form the kingdom. Each of the next **n** lines will contain two integers: **x_i y_i ($-10^5 \leq x_i, y_i \leq 10^5$)** denoting the *i*th vertex of the polygon. The vertices will be given in anti-clockwise order.

The next line will contain an integer **Q ($1 \leq Q \leq 10000$)** denoting the number of positions reported by the messengers. Each of the next **Q** lines will contain two integers: **x_i y_i ($-10^5 \leq x_i, y_i \leq 10^5$)** denoting the position of the *i*th person.

Output

For each case, print the case number in a single line. Then for each reported person, print 'y' or 'n' depending whether the person is an inhabitant or not.

Sample Input	Output for Sample Input
1 3 0 0 10 0 5 5 3 1 2 5 6 4 3	Case 1: n n y

Note

Dataset is huge. Use faster i/o methods.

1197 – Help Hanzo

Amakusa, the evil spiritual leader has captured the beautiful princess Nakururu. The reason behind this is he had a little problem with Hanzo Hattori, the best ninja and the love of Nakururu. After hearing the news Hanzo got extremely angry. But he is clever and smart, so, he kept himself cool and made a plan to face Amakusa.

Before reaching Amakusa's castle, Hanzo has to pass some territories. The territories are numbered as **a, a+1, a+2, a+3 ... b**. But not all the territories are safe for Hanzo because there can be other fighters waiting for him. Actually he is not afraid of them, but as he is facing Amakusa, he has to save his stamina as much as possible.

He calculated that the territories which are primes are safe for him. Now given **a** and **b** he needs to know how many territories are safe for him. But he is busy with other plans, so he hired you to solve this small problem!

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case contains a line containing two integers **a** and **b ($1 \leq a \leq b < 2^{31}$, $b - a \leq 100000$)**.

Output

For each case, print the case number and the number of safe territories.

Sample Input	Output for Sample Input
3 2 36 3 73 3 11	Case 1: 11 Case 2: 20 Case 3: 4

Note

A number is said to be prime if it is divisible by exactly two different integers. So, first few primes are 2, 3, 5, 7, 11, 13, 17, ...

1198 - Karate Competition

Your karate club challenged another karate club in your town. Each club enters **N** players into the match, and each player plays one game against a player from the other team. Each game that is won is worth **2** points, and each game that is drawn is worth **1** point. Your goal is to score as many points as possible.

Your secret agents have determined the skill of every member of the opposing team, and of course you know the skill of every member of your own team. You can use this information to decide which opposing player will play against each of your players in order to maximize your score. Assume that the player with the higher skill in a game will always win, and if the players have the same skill then they will draw.

You will be given the skills of your players and of the opposing players, you have to find the maximum number of points that your team can score.

Input

Input starts with an integer **T** (≤ 70), denoting the number of test cases.

Each case starts with a line containing an integer **N** ($1 \leq N \leq 50$). The next line contains **N** space separated integers denoting the skills of the players of your team. The next line also contains **N** space separated integers denoting the skills of the players of the opposite team. Each of the skills lies in the range [1, 1000].

Output

For each case, print the case number and the maximum number of points your team can score.

Sample Input	Output for Sample Input
4 2 4 7 6 2 2 6 2 4 7 3 5 10 1 5 10 1 4 10 7 1 4 15 3 8 7	Case 1: 4 Case 2: 2 Case 3: 4 Case 4: 5

1199 – Partitioning Game

Alice and Bob are playing a strange game. The rules of the game are:

1. Initially there are **n** piles.
2. A pile is formed by some cells.
3. Alice starts the game and they alternate turns.
4. In each turn a player can pick any pile and divide it into two unequal piles.
5. If a player cannot do so, he/she loses the game.

Now you are given the number of cells in each of the piles, you have to find the winner of the game if both of them play optimally.

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 100$). The next line contains **n** integers, where the i^{th} integer denotes the number of cells in the i^{th} pile. You can assume that the number of cells in each pile is between 1 and **10000**.

Output

For each case, print the case number and '**Alice**' or '**Bob**' depending on the winner of the game.

Sample Input	Output for Sample Input
3 1 4 3 1 2 3 1 7	Case 1: Bob Case 2: Alice Case 3: Bob

Explanation

In case 1, Alice has only 1 move, she divides the pile with 4 cells into two unequal piles, where one pile has 1 cell and the other pile has 3 cells. Now it's Bob's turn. Bob divides the pile with 3 cells into two piles, where one pile has 1 cell and another pile has 2 cells. So, now there are three piles having cells 1, 1, 2. And Alice loses, since she doesn't have any moves now.

1200 – Thief

A thief has entered into a super shop at midnight. Poor thief came here because his wife has sent him to buy some necessary households. Instead of buying the items, he decided to steal them.

He has a bag with him which can carry up to W kg. In the list (given by his wife) there are four fields 1) item name, 2) the price of the item, 3) how many of this item is required and 4) the weight of this item. The items are solid items, so he can't take any item after dividing into pieces.

Now the thief wants to take items in his bag such that it fulfills his wife's list, and the total weight of the items is not greater than W . And he can't take any item other than the items mentioned in the list, otherwise his wife would find the item and he may get caught. But he can take more items than required. And he wants to sell these extra items and earn some money. Assume that he can take any item (is in the list) any number of times unless they overflow his bag.

Now you are given the necessary information of the items and his bag, you have to find the maximum profit the thief can make.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 100$) and W ($1 \leq W \leq 10000$), where n denotes the number of items. Each of the next n lines contains three integers $p_i \ c_i \ w_i$ ($1 \leq p_i, c_i, w_i \leq 100$), meaning that the price of the i^{th} item is p_i , c_i of it must be taken, and the weight is w_i .

Output

For each case, print the case number and the maximum profit he can get. If it's impossible to fulfill his wife's requirements, print '**Impossible**'.

Sample Input	Output for Sample Input
2 3 20 10 1 10 5 1 5 1 1 1 1 10 10 1 11	Case 1: 4 Case 2: Impossible

1201 – A Perfect Murder

"Yes, I am the murderer. No doubt" I had to confess it in front of all. But wait, why I am confessing? Nobody wants to go to jail, neither do I. As you have suspected there is something fishy. So, let me explain a bit.

The murder was happened in 19th June, at 11:30 pm this year (2009) according to the medical report. So, I was asking the judge "Can you find the time 19th June 11:30 pm in Bangladesh?" The judge informed other reporters to find the time. But alas! There was no time - "2009, 19th June, 11:30 pm". So, the judge got a bit confused about my confession. So, I began to tell them, "The time the murder was happened, is not a valid time according to you. So, how can you claim that I am the murderer?"

And what happened next, you all know. I am in the streets again with a clean sheet.

But now I have planned to kill again. I have a list of **N** mosquitoes which are to be killed. But there is a small problem. If I kill a mosquito, all of his friends will be informed, so they will be prepared for my attack, thus they will be impossible to kill. But there is a surprising fact. That is if I denote them as a node and their friendship relations as edges, the graph becomes acyclic.

Now I am planning when and how to kill them (how to get rid of the law!) and you have to write a program that will help me to find the maximum number of mosquito I can kill. Don't worry too much, if anything goes wrong I will not mention your name, trust me!

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a blank line and two integers **N ($1 \leq N \leq 1000$)** denoting the number of mosquito I want to kill and **M** denoting the number of friendship configurations. Each of the next **M** lines contains two integers **a** and **b** denoting that a^{th} and b^{th} mosquitoes are friends. You can assume that **($1 \leq a, b \leq N, a \neq b$)** and each friendship relation is given only once. As I have already mentioned, you will not find any cycle in the relations.

Output

For each case, print the case number and the maximum number of mosquitoes I can kill considering the conditions described above.

Sample Input	Output for Sample Input
<p>3</p> <p>4 3</p> <p>1 2</p> <p>1 3</p> <p>1 4</p> <p>3 2</p> <p>1 2</p> <p>2 3</p> <p>5 4</p> <p>1 2</p> <p>1 3</p> <p>2 4</p> <p>2 5</p>	<p>Case 1: 3</p> <p>Case 2: 2</p> <p>Case 3: 3</p>

1202 - Bishops

There is an Infinite chessboard. Two bishops are there. (Bishop means the chess piece that moves diagonally).

Now you are given the position of the two bishops. You have to find the minimum chess moves to take one to another. With a chess move, a bishop can be moved to a long distance (along the diagonal lines) with just one move.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case contains four integers r_1 c_1 r_2 c_2 denoting the positions of the bishops. Each of the integers will be positive and not greater than 10^9 . You can also assume that the positions will be distinct.

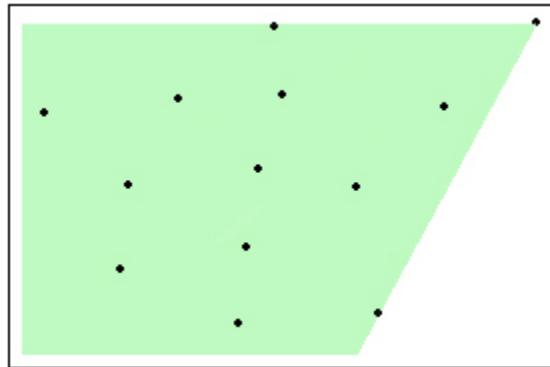
Output

For each case, print the case number and the minimum moves required to take one bishop to the other. Print '**impossible**' if it's not possible.

Sample Input	Output for Sample Input
3 1 1 10 10 1 1 10 11 1 1 5 3	Case 1: 1 Case 2: impossible Case 3: 2

1203 – Guarding Bananas

Once there was a lazy monkey in a forest. But he loved banana too much. One day there was a storm in the jungle and all the bananas fell from the trees. The monkey didn't want to lose any of the bananas. So, he wanted to find a banana such that he can eat that and he can also look after the other bananas. As he was lazy, he didn't want to move his eyes too wide. So, you have to help him finding the banana from where he can look after all the bananas but the degree of rotating his eyes is as small as possible. You can assume that the position of the bananas can be modeled as 2D points.



Here a banana is shown, from where the monkey can look after all the bananas with minimum eye rotation.

Input

Input starts with an integer **T** (≤ 13), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 10^5$) denoting the number of bananas. Each of the next **n** lines contains two integers **x y** ($-10^9 \leq x, y \leq 10^9$) denoting the co-ordinate of a banana. There can me more than one bananas in the same co-ordinate.

Output

For each case, print the case number and the minimum angle in degrees. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 1 4 4 4 0 0 10 0 10 10 2 1	Case 1: 0 Case 2: 45.000000

Note

Dataset is huge. Use faster I/O methods.

1204 - Weird Advertisement

Renat Mulkhanov (**rem**), one of the most talented programmers in the world, passed away on March 11, 2011. This is very sad news for all of us. His team went to ACM ICPC World Finals - 2004, placed 4th and won gold medals. He really was a great programmer. May he rest in peace. This problem is dedicated to him.

2DPlaneLand is a land just like a huge **2D** plane. The range of **X** axis is **0** to **10^9** and the range of **Y** axis is also **0** to **10^9** . People built houses only in integer co-ordinates and there is exactly one house in each integer co-ordinate.

Now **UseAndSmile** Soap Company is launching a new soap. That's why they want to advertise this product as much as possible. So, they selected **n** persons for this task. Each person will be given a rectangular region. He will advertise the product to all the houses that lie in his region. Each rectangular region is identified by 4 integers **x₁, y₁, x₂ and y₂**. That means this person will advertise in all the houses whose **x** co-ordinate is between **x₁** and **x₂** (inclusive) and **y** co-ordinate is between **y₁** and **y₂** (inclusive).

Now after a while they realized that some houses are being advertised by more than one person. So, they want to find the number of houses that are advertised by at least **k** persons. Since you are one of the best programmers in the city; they asked you to solve this problem.

Input

Input starts with an integer **T** (≤ 13), denoting the number of test cases.

Each case starts with a line containing two integers **n** ($1 \leq n \leq 30000$), **k** ($1 \leq k \leq 10$). Each of the next **n** lines will contain four integers **x₁, y₁, x₂, y₂** ($0 \leq x_1, y_1, x_2, y_2 \leq 10^9$, $x_1 < x_2$, $y_1 < y_2$) denoting a rectangular region for a person.

Output

For each case, print the case number and the total number of houses that are advertised by at least **k** people.

Sample Input	Output for Sample Input
2 2 1 0 0 4 4 1 1 2 5 2 2 0 0 4 4 1 1 2 5	Case 1: 27 Case 2: 8

Note

Dataset is huge. Use faster i/o methods.

1205 – Palindromic Numbers

A palindromic number or numeral palindrome is a 'symmetrical' number like 16461 that remains the same when its digits are reversed. In this problem you will be given two integers i j , you have to find the number of palindromic numbers between i and j (inclusive).

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case starts with a line containing two integers i j ($0 \leq i, j \leq 10^{17}$).

Output

For each case, print the case number and the total number of palindromic numbers between i and j (inclusive).

Sample Input	Output for Sample Input
4 1 10 100 1 1 1000 1 10000	Case 1: 9 Case 2: 18 Case 3: 108 Case 4: 198

1206 - Scheduling Taxi Cabs

Your taxi cab company is one of the best taxi cab networks in town. But running a taxi station is definitely not simple. Apart from the obvious demand for a centralized coordination of the cabs in order to pick up the customers calling to get a cab as soon as possible, there is also a need to schedule all the taxi rides which have been booked in advance. Given a list of all booked taxi rides for the next day, you want to minimize the number of cabs needed to carry out all of the rides.

For simplicity, we model a city as a rectangular grid. An address in the city is denoted by two integers: the street and avenue number. The time needed to get from the address **(a, b)** to **(c, d)** by taxi is $|a - c| + |b - d|$ minutes. A cab may carry out a booked ride if it is its first ride of the day, or if it can get to the source address of the new ride from its latest, at least one minute before the new ride's scheduled departure. Note that some rides may end after midnight.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing an integer **M ($0 < M < 500$)**, being the number of booked taxi rides. The following **M** lines contain the rides. Each ride is described by a departure time on the format **hh:mm** (ranging from **00:00** to **23:59**), two integers **a b** that are the coordinates of the source address and two integers **c d** that are the coordinates of the destination address. All coordinates are at least **0** and strictly smaller than **200**. The booked rides in each scenario are sorted in order of increasing departure time.

Output

For each case, print the case number and the minimum number of cabs required to carry out all the booked taxi rides.

Sample Input	Output for Sample Input
2 2 08:00 10 11 9 16 08:06 9 16 10 11 2 08:00 10 11 9 16 08:07 9 16 10 11	Case 1: 2 Case 2: 1

1207 - Posters for Election

Election Day has arrived in Bangladesh. The citizens are aware of the fact that the candidates will poster the entire country, ruining the walls, pillars and windows. Thus they have decided to limit the entire election campaign to a single dedicated wall. Every candidate will be allowed to hang exactly one poster on the wall. All posters extend from top to bottom, but are hung at different points of the wall, and may be of different width. The wall is divided horizontally into sections, and a poster completely occupies one or more adjacent sections.

But the candidates used the wall without considering the posters from other candidates. Some of the posters were covered (partially or completely) by those of other candidates. Now you are given the location of all the posters and the order in which they were hung, determine how many posters have at least one visible section in the end.

Input

Input starts with an integer T (≤ 8), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 10^5$) denoting the number of posters. Each of the next n lines contains two integers l_i , r_i ($1 \leq l_i \leq r_i \leq 2*n$) denoting the number of the wall section occupied by the left end and the right end of the i^{th} poster, respectively. The input order corresponds to the order of hanging posters.

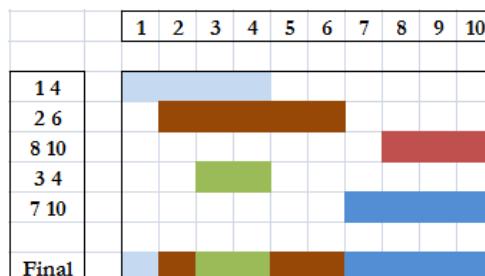
Output

For each case, print the case number and the number of posters with visible sections.

Sample Input	Output for Sample Input
1 5 1 4 2 6 8 10 3 4 7 10	Case 1: 4

Note

1. Dataset is huge. Use faster I/O methods.
2. Illustration for sample input:



1208 - Dangerous Bull! Who Wants to Pull?

A mad dangerous bull has freed himself from its chain. Now it's attacking all people that come by in its path. So, people of village 'Goru Mari' are frightened and at a loss. One of the villagers reported that the bull is sitting on a place (actually after all days hard work, the bull was resting). So, the villagers made a plan to imprison the cow with a fence.

Assume the village as a 2D grid, where the bull is sitting on coordinate (x, y) . And some pairs of coordinates will be given $(x_1, y_1), (x_2, y_2)$, that means the villagers can put bamboos between this two points. Actually there is a tree in (x_1, y_1) and also one in (x_2, y_2) , and there are branches in the trees such that the villagers can tie **two** bamboos so that the bull won't be able to cross. The villagers are not able to place bamboos between trees which are not listed (may be, there are no suitable branches in those trees such that a bamboo can be tied).

Now the villagers want to imprison the bull, that means they want to use some bamboos to cover an area such that the bull will be inside that area and not able to cross that area. The villagers want the area to be convex. That means each angle is less than or equal to 180° . If there is a bamboo position that crosses any other bamboo position, then only one of them can be used.

Since bamboos are not so cheap, so the villagers want to imprison the bull such that the total length of the bamboo is as small as possible. And villagers are not good at math, so they asked your help.

Input

Input starts with an integer T (≤ 125), denoting the number of test cases.

Each case starts with a blank line. The next line contains three integers $n x y$ where n ($1 \leq n \leq 100$) denotes the number of tree pairs, (x, y) means the position of the bull. Each of the next n lines contains four integers $x_1 y_1 x_2 y_2$, meaning that you can place bamboos between (x_1, y_1) and (x_2, y_2) . You can assume that all the co-ordinates given for this problem satisfy $(-10^4 \leq x_i, y_i \leq 10^4)$ and no two trees and the position of the bull will be collinear.

Output

For each case, print the case number and the minimum length of the bamboos needed. If no such solution is found, print **-1**. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 3 2 1 0 0 10 0 10 0 0 10 0 10 0 0 1 5 5 0 0 10 0	Case 1: 68.2842712475 Case 2: -1.000

1209 - Strange Voting

Its year 3000, and the voting system in Ajobdesh has changed to a new era. Instead of the boring old style voting, the new style voting is applied as follows:

1. Initially there are m male candidates and f female candidates for the parliament. For simplicity the male candidates are numbered as ' M_1 ', ' M_2 ' ... ' M_m ' and the female candidates are numbered as ' F_1 ', ' F_2 ' ... ' F_f '.
2. There are v voters, and each of them can vote like ' $P Q$ ', which means, he wants to see P in the parliament and he wants Q to be thrown out of the parliament. For example, if a person voted like, ' $M_3 F_7$ ', that means he wants M_3 to be elected and F_7 to be thrown out.
3. The parliament will be formed in such a way that the maximum number of votes is satisfied. A voter, who voted ' $P Q$ ', is said to be satisfied if P is in the parliament and Q is not. For example, let the parliament be $\{M_1, M_3, F_3\}$, then voter who voted ' $M_1 F_4$ ' is satisfied but neither of ' $M_1 F_3$ ', ' $F_3 M_3$ ' or ' $M_2 F_3$ ' is satisfied.

Since Men don't want to see any Woman in the parliament, each man always votes like ' $M_x F_y$ '. And Women don't want Men in the parliament, so each woman always votes like ' $F_y M_x$ '. Assume that only Men and Women are eligible for voting.

Since you are the leading programmer in Ajobdesh, you have to form the parliament such that maximum number of voters is satisfied. Just report the maximum number of satisfied voters.

Input

Input starts with an integer T (≤ 25), denoting the number of test cases.

Each case starts with a line containing three integers: m, f, v ($1 \leq m, f \leq 100, 0 \leq v \leq 500$). Each of the next v lines contains a vote either in the form ' $M_x F_y$ ' or ' $F_y M_x$ ' ($1 \leq x \leq m, 1 \leq y \leq f$).

Output

For each case, print the case number and the maximum number of satisfied voters.

Sample Input	Output for Sample Input
2 1 1 2 M1 F1 F1 M1 1 2 5 M1 F1 M1 F1 M1 F2 F2 M1 F1 M1	Case 1: 1 Case 2: 3

1210 – Efficient Traffic System

I was given the task to make all the major two way roads in Bangladesh into one way roads. And I have done that easily with some great pruning. And I asked the Govt. Traffic Management System to change the direction of all the roads.

But after some days, I realized that I should have thought of the fact that all cities should be reachable from other cities using the existing one way roads. Since the traffic system is already designed, so it may not be changed. But I can ask the govt. to build new roads between any pair of cities.

Now since the task looks quite hard for me, I am asking you to do it for me. I will give you the current roads configuration. You have to find the minimum number of roads that have to be built such that it's possible to go from any city to any other city.

Input

Input starts with an integer **T (≤ 25)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n ($1 \leq n \leq 20000$)** and **m ($0 \leq m \leq 50000$)**, where **n** denotes the number of cities and **m** denotes the number of one way roads. Each of the next **m** lines contains two integers **u v ($1 \leq u, v \leq n, u \neq v$)** meaning that there is a road from **u** to **v**. Assume that there can be at most one road from a city **u** to **v**.

Output

For each case, print the case number and the minimum number of roads that have to be built.

Sample Input	Output for Sample Input
2 3 0 3 2 1 2 1 3	Case 1: 3 Case 2: 2

Note

Dataset is huge. Use faster I/O methods.

1211 - Intersection of Cubes

You are given n cubes, each cube is described by two points in 3D space: (x_1, y_1, z_1) being one corner of the cube and (x_2, y_2, z_2) being the opposite corner. Assume that the sides of each of the cubes are parallel to the axis. Your task is to find the volume of their intersection.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 100$). Each of the next n lines contains six integers $x_1 \ y_1 \ z_1 \ x_2 \ y_2 \ z_2$ ($1 \leq x_1, y_1, z_1, x_2, y_2, z_2 \leq 1000, x_1 < x_2, y_1 < y_2, z_1 < z_2$) where (x_1, y_1, z_1) is the co-ordinate of one corner and (x_2, y_2, z_2) is the co-ordinate of the opposite corner.

Output

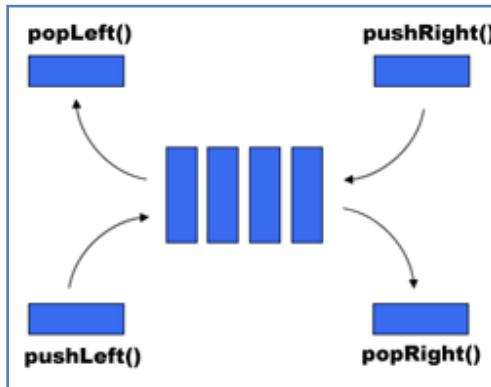
For each case, print the case number and volume of their intersection.

Sample Input	Output for Sample Input
2 2 1 1 1 3 3 3 1 1 1 2 2 2 3 7 8 9 20 20 30 2 2 2 50 50 50 13 14 15 18 30 40	Case 1: 1 Case 2: 450

1212 - Double Ended Queue

A queue is a data structure based on the principle of 'First In First Out' (FIFO). There are two ends; one end can be used only to insert an item and the other end to remove an item. A Double Ended Queue is a queue where you can insert an item in both sides as well as you can delete an item from either side. There are mainly four operations available to a double ended queue. They are:

1. **pushLeft()**: inserts an item to the left end of the queue with the exception that the queue is not full.
2. **pushRight()**: inserts an item to the right end of the queue with the exception that the queue is not full.
3. **popLeft()**: removes an item from the left end of the queue with the exception that the queue is not empty.
4. **popRight()**: removes an item from the right end of the queue with the exception that the queue is not empty.



Now you are given a queue and a list of commands, you have to report the behavior of the queue.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing two integers **n, m ($1 \leq n \leq 10, 1 \leq m \leq 100$)**, where **n** denotes the size of the queue and **m** denotes the number of commands. Each of the next **m** lines contains a command which is one of:

pushLeft x	pushes x ($-100 \leq x \leq 100$) in the left end of the queue
pushRight x	pushes x ($-100 \leq x \leq 100$) in the right end of the queue
popLeft	pops an item from the left end of the queue
popRight	pops an item from the right end of the queue

Output

For each case, print the case number in a line. Then for each operation, show its corresponding output as shown in the sample. Be careful about spelling.

Sample Input	Output for Sample Input
<pre> 1 3 8 pushLeft 1 pushLeft 2 pushRight -1 pushRight 1 popLeft popRight popLeft popRight </pre>	<pre> Case 1: Pushed in left: 1 Pushed in left: 2 Pushed in right: -1 The queue is full Popped from left: 2 Popped from right: -1 Popped from left: 1 The queue is empty </pre>

1213 – Fantasy of a Summation

If you think codes, eat codes then sometimes you may get stressed. In your dreams you may see huge codes, as I have seen once. Here is the code I saw in my dream.

```
#include <stdio.h>

int cases, caseno;
int n, K, MOD;
int A[1001];

int main() {
    scanf("%d", &cases);
    while( cases-- ) {
        scanf("%d %d %d", &n, &K, &MOD);

        int i, i1, i2, i3, ... , iK;

        for( i = 0; i < n; i++ ) scanf("%d", &A[i]);

        int res = 0;
        for( i1 = 0; i1 < n; i1++ ) {
            for( i2 = 0; i2 < n; i2++ ) {
                for( i3 = 0; i3 < n; i3++ ) {
                    ...
                    for( iK = 0; iK < n; iK++ ) {
                        res = ( res + A[i1] + A[i2] + ... + A[iK] ) % MOD;
                    }
                    ...
                }
            }
        }
        printf("Case %d: %d\n", ++caseno, res);
    }
    return 0;
}
```

Actually the code was about: 'You are given three integers **n**, **K**, **MOD** and **n** integers: **A₀**, **A₁**, **A₂** ... **A_{n-1}**, you have to write **K** nested loops and calculate the summation of all **A_i** where **i** is the value of any nested loop variable.'

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with three integers: **n** ($1 \leq n \leq 1000$), **K** ($1 \leq K < 2^{31}$), **MOD** ($1 \leq MOD \leq 35000$). The next line contains **n** non-negative integers denoting **A₀**, **A₁**, **A₂** ... **A_{n-1}**. Each of these integers will be fit into a 32 bit signed integer.

Output

For each case, print the case number and result of the code.

Sample Input	Output for Sample Input
2 3 1 35000 1 2 3 2 3 35000 1 2	Case 1: 6 Case 2: 36

1214 - Large Division

Given two integers, **a** and **b**, you should check whether **a** is divisible by **b** or not. We know that an integer **a** is divisible by an integer **b** if and only if there exists an integer **c** such that **a = b * c**.

Input

Input starts with an integer **T** (≤ 525), denoting the number of test cases.

Each case starts with a line containing two integers **a** ($-10^{200} \leq a \leq 10^{200}$) and **b** ($|b| > 0$, **b** fits into a 32 bit signed integer). Numbers will not contain leading zeroes.

Output

For each case, print the case number first. Then print '**divisible**' if **a** is divisible by **b**. Otherwise print '**not divisible**'.

Sample Input	Output for Sample Input
6 101 101 0 67 -101 101 7678123668327637674887634 101 11010000000000000000 256 -202202202202000202202202 -101	Case 1: divisible Case 2: divisible Case 3: divisible Case 4: not divisible Case 5: divisible Case 6: divisible

1215 – Finding LCM

LCM is an abbreviation used for Least Common Multiple in Mathematics. We say **LCM (a, b, c) = L** if and only if **L** is the least integer which is divisible by **a, b** and **c**.

You will be given **a, b** and **L**. You have to find **c** such that **LCM (a, b, c) = L**. If there are several solutions, print the one where **c** is as small as possible. If there is no solution, report so.

Input

Input starts with an integer **T (≤ 325)**, denoting the number of test cases.

Each case starts with a line containing three integers **a b L ($1 \leq a, b \leq 10^6, 1 \leq L \leq 10^{12}$)**.

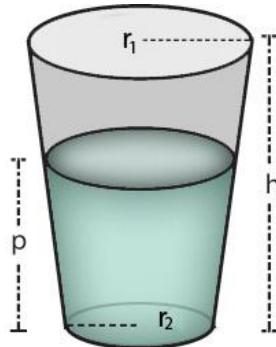
Output

For each case, print the case number and the minimum possible value of **c**. If no solution is found, print '**impossible**'.

Sample Input	Output for Sample Input
3 3 5 30 209475 6992 77086800 2 6 10	Case 1: 2 Case 2: 1 Case 3: impossible

1216 – Juice in the Glass

Once upon a time, there lived a mad programmer. He loved to solve creative problems other than anything. His wife loved him quite a lot but disliked his curiosity for the problems. One day he came from office, his wife gave him a glass of cold lime juice. She was in a romantic mood and waiting for some romantic stuff. But the programmer asked her curiously, "If I give u radius of the top and bottom part of the glass and the height, can you come up with the volume of the glass?" His wife became a bit disappointed but as she is smart she replied with a smile, "You already have drunk some juice, and the glass is not full. If I give you the height of the juice, can you find the volume of the remaining juice in the glass?" Then the programmer kissed his wife and said, "You are the best problem setter in the world!"



Now he set the same problem for you. The radius of the upper part r_1 and lower part r_2 is given. If height of the glass is h and height of the juice is p what is the volume of the juice in the glass?

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing four integers $r_1 \ r_2 \ h \ p$ ($1 \leq r_2 < r_1 \leq 100, 1 \leq p \leq h \leq 100$).

Output

For each case, print the case number and the volume of the juice in the glass. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 5 2 3 2 5 2 3 3	Case 1: 58.643062867 Case 2: 122.52211349

1217 - Neighbor House (II)

A soap company wants to advertise their product in a local area. In this area, there are **n** houses and the houses are placed in circular fashion, such that house **1** has two neighbors: house **2** and **n**. House **5** has two neighbors: house **4** and **6**. House **n** has two neighbors, house **n-1** and **1**.

Now the soap company has an estimation of the number of soaps they can sell on each house. But for their advertising policy, if they sell soaps to a house, they can't sell soaps to its two neighboring houses. No your task is to find the maximum number of estimated soaps they can sell in that area.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($2 \leq n \leq 1000$). The next line contains **n** space separated integers, where the **ith** integer denotes the estimated number of soaps that can be sold to the **ith** house. Each of these integers will lie in the range [1, 1000].

Output

For each case, print the case number and the maximum number of estimated soaps that can be sold in that area.

Sample Input	Output for Sample Input
3 2 10 100 3 10 2 11 4 8 9 2 8	Case 1: 100 Case 2: 11 Case 3: 17

1218 – Multiple Free Subset

Given a set of n integers, you have to take a subset such that the set is multiple free. That means you have to take a subset such that if you pick any two elements p, q from the subset, then p is not a multiple of q or q is not a multiple of p .

For example, let the set be $\{2, 5, 10, 8\}$, then your subset can be $\{2\}$ or $\{10\}$ or $\{5, 8\}$, etc. But you can't take $\{2, 8\}$ since 8 is multiple of 2, or you can't take $\{10, 5\}$ since 10 is multiple of 5.

Now your task is to find such a subset with maximum number of elements. There can be several solutions with maximum number of elements. In such case, we break the tie by the following:

Let $\{a_1, a_2, a_3 \dots a_m\}$ be a solution and $\{b_1, b_2, b_3 \dots b_m\}$ be another solution where $a_i < a_{i+1}$ and $b_i < b_{i+1}$ for $i = 1$ to $m-1$. Then $\{a_1, a_2, a_3 \dots a_m\}$ is the result if

$a_i < b_i$ or

$a_i = b_i$ and $a_{i+1} < b_{i+1}$ or

$a_i = b_i$ and $a_{i+1} = b_{i+1}$ and $a_{i+2} < b_{i+2}$ or

...

For example, let one solution be $\{3, 5\}$ and another solution be $\{3, 10\}$, then we want $\{3, 5\}$ as our result.

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 100$). The next line contains n space separated integers forming the set. Each of these integers will lie in the range $[1, 10^9]$.

Output

For each case, print the case number and the multiple free subset as stated above.

Sample Input	Output for Sample Input
6 3 2 2 4 4 2 4 6 10 6 7 1 2 2 5 9 4 3 10 10 5 5 7 2 9 9 49 10 1 2 3 4 5 6 7 8 9 10	Case 1: 2 Case 2: 4 6 10 Case 3: 2 5 7 9 Case 4: 3 5 Case 5: 2 7 9 Case 6: 4 5 6 7 9

1219 – Mafia

Don Seliari gives his right hand Tommas Angelo (Tommy) an important job to guard his territory. The mafia territory consists of **n** cities. And the cities are connected to each other in such a way that there is exactly one path from any city to another. Tommy has exactly **n** mafia boys to guard the cities. Initially the mafia boys are resting randomly at the **n** cities. Tommy wants every city to be guarded by the mafia boys. This is to be accomplished by a sequence of moves; each move consists of moving one mafia boy to the adjacent city. What is the minimum number of moves required so that every city is guarded?

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 10000$). Each of the next **n** line contains at least three numbers which are: **v** the number of a city, followed by the number of mafia boys placed at city **v** followed by a number **d** which is the number of cities adjacent of **v**, followed by **d** numbers giving the adjacent cities of **v**. Amongst the cities, one city is chosen as root, and for any city all its adjacent cities are listed except the one that connects it (through a path) to the root city.

Output

For each case, print the case number and the minimum number of moves required for the mafia boys to guard all cities.

Sample Input	Output for Sample Input
2 9 1 2 3 2 3 4 2 1 0 3 0 2 5 6 4 1 3 7 8 9 5 3 0 6 0 0 7 0 0 8 2 0 9 0 0 9 1 0 3 2 3 4 2 0 0 3 0 2 5 6 4 9 3 7 8 9 5 0 0 6 0 0 7 0 0 8 0 0 9 0 0	Case 1: 7 Case 2: 14

1220 – Mysterious Bacteria

Dr. Mob has just discovered a Deathly Bacteria. He named it RC-01. RC-01 has a very strange reproduction system. RC-01 lives exactly x days. Now RC-01 produces exactly p new deadly Bacteria where $x = b^p$ (where b, p are integers). More generally, x is a perfect p^{th} power. Given the lifetime x of a mother RC-01 you are to determine the maximum number of new RC-01 which can be produced by the mother RC-01.

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a line containing an integer x . You can assume that x will have magnitude at least 2 and be within the range of a 32 bit signed integer.

Output

For each case, print the case number and the largest integer p such that x is a perfect p^{th} power.

Sample Input	Output for Sample Input
3 17 1073741824 25	Case 1: 1 Case 2: 30 Case 3: 2

1221 - Travel Company

A travel company is planning to launch their bus service in a new route. So they conducted a survey and made a list of all possible roads connecting different cities. Each of the roads has a certain amount of income based on current fare. But at the same time, each road has some expenses too (this includes fuel and maintenance cost, staff payments, taxes and tribute to labor union which is recently approved by the Government). The travel company is looking for a cyclic route. That is, the bus will start from any city, then visit one or more cities each exactly once and return to the starting city. The company is also concerned with the profit on the route. In fact the directors of the company have a strict requirement of a profit ratio strictly greater than P . Otherwise they will not launch the service. A profit ratio for a route is the ratio between the total incomes to the total expenses for that route.

One of your friends works in that company and he asks for a little help from you. All you have to do is to determine if there exists such route, so that the company has a profit ratio of P .

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a blank line and three integers N, R, P ($2 \leq N \leq 100, 0 \leq R \leq 9900, 1 \leq P \leq 100$). N , R and P represents number of cities, number of road links and the expected profit ratio respectively. Then R lines follow. Each line contains four integers A_i, B_i, I_i, E_i ($0 \leq A_i, B_i < N, 0 \leq I_i \leq 5000, 1 \leq E_i \leq 5000$). (A_i, B_i) represents directed road link from city A_i to B_i . I_i and E_i are the incomes and expenses of the road link respectively. You may assume that $(A_i, B_i) \neq (A_j, B_j)$, if $i \neq j$ and $A_i \neq B_i$ for any i .

Output

For each case, print the case number and "YES" if there is a cyclic route for which the profit ratio is greater than P or "NO", if there is no such route.

Sample Input	Output for Sample Input
<pre> 3 5 8 3 0 1 17 8 1 0 10 5 1 2 11 5 1 4 5 3 2 3 13 7 3 1 9 4 4 3 11 1 3 0 11 6 5 8 3 0 1 17 8 1 0 10 5 1 2 11 5 1 4 5 3 2 3 13 7 3 1 9 4 4 3 11 2 3 0 11 6 5 8 2 0 1 17 8 1 0 10 5 1 2 11 5 1 4 5 3 2 3 13 7 3 1 9 4 4 3 11 5 3 0 11 6 </pre>	Case 1: YES Case 2: NO Case 3: YES

Note

Dataset is huge. Use faster I/O methods.

1222 – Gift Packing

There are **n** gifts and **n** boxes, each box can contain at most one gift. Now you want to pack all the gifts in the boxes such that your profit is as high possible.

The boxes are numbered from **1** to **n** and the gifts are numbered from **1** to **n**. You will be given an $(n \times n)$ matrix where p_{ij} denotes the profit if we put the i^{th} gift into the j^{th} box. Now your task is to pack all the gifts and maximize the profit.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 50$). Each of the next **n** lines contains **n** space separated integers forming the matrix. The values in the matrix lie in the range **[0, 1000]**.

Output

For each case, print the case number and the maximum profit.

Sample Input	Output for Sample Input
2 2 4 3 3 1 3 1 4 5 5 7 6 5 8 8	Case 1: 6 Case 2: 18

1223 - Testing Mailboxes

When monkeys are given some fire-crackers, they have only thing in the mind - to blow things up. Small boxes were easy to blow up, and thus mailboxes became a popular target. Now, a small mailbox manufacturer is interested in how many fire-crackers his new mailbox prototype can withstand without exploding and has hired you to help him. He will provide you with k identical mailbox prototypes each fitting up to m fire-crackers. However, he is not sure of how many fire-crackers he needs to provide you with in order for you to be able to solve his problem, so he asks your help.

The constraints are:

1. If you blow up a mailbox, you can't use the mailbox again, so if you have only $k = 1$ mailboxes, you would have to start testing with 1 fire-cracker, then 2 fire-crackers, and so on until it finally exploded. In the worst case, that is if it does not blow up even when filled with m fire-crackers, you would need $1 + 2 + 3 + \dots + m = m * (m + 1)/2$ fire-crackers.
2. If a mailbox can withstand x fire-crackers, it can also withstand $x-1$ fire-crackers.
3. Upon an explosion, a mailbox is either totally destroyed (blown up) or unharmed, which means that it can be reused in another test explosion.

Now the manufacturer wants you to find the maximum number of fire-crackers that his mailboxes can withstand. Before doing that you have to buy some fire-crackers to test that. So, you need to find the minimum number of fire-crackers you need to buy to test the mailboxes.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case starts with a line containing two integers: k ($1 \leq k \leq 100$) and m ($1 \leq m \leq 100$).

Output

For each case, print the case number and the minimum number of fire-crackers you have to buy.

Sample Input	Output for Sample Input
4 1 10 3 73 5 100 1 100	Case 1: 55 Case 2: 382 Case 3: 495 Case 4: 5050

1224 - DNA Prefix

Given a set of **n** DNA samples, where each sample is a string containing characters from {**A, C, G, T**}, we are trying to find a subset of samples in the set, where the length of the longest common prefix multiplied by the number of samples in that subset is maximum.

To be specific, let the samples be:

```
ACGT
ACGTGCGT
ACCGTGC
ACGCCGT
```

If we take the subset {ACGT} then the result is 4 ($4 * 1$), if we take {ACGT, ACGTGCGT, ACGCCGT} then the result is $3 * 3 = 9$ (since ACG is the common prefix), if we take {ACGT, ACGTGCGT, ACCGTGC, ACGCCGT} then the result is $2 * 4 = 8$.

Now your task is to report the maximum result we can get from the samples.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 50000$) denoting the number of DNA samples. Each of the next **n** lines contains a non empty string whose length is not greater than **50**. And the strings contain characters from {**A, C, G, T**}.

Output

For each case, print the case number and the maximum result that can be obtained.

Sample Input	Output for Sample Input
3 4 ACGT ACGTGCGT ACCGTGC ACGCCGT 3 CGCGCGCGCGCGCCCCGCCCGCGC CGCGCGCGCGCGCCCCGCCCGCAC CGCGCGCGCGCGCCCCGCCCGCTC 2 CGCGCCCGCGCGCGCGCGCGC GGCGCCCGCGCGCGCGCGCTC	Case 1: 9 Case 2: 66 Case 3: 20

Note

Dataset is huge. Use faster I/O methods.

1225 - Palindromic Numbers (II)

A palindromic number or numeral palindrome is a 'symmetrical' number like 16461, that remains the same when its digits are reversed. In this problem you will be given an integer, you have to say whether the number is a palindromic number or not.

Input

Input starts with an integer **T** (≤ 20000), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($0 \leq n < 10^9$).

Output

For each case, print the case number and '**Yes**' if **n** is palindromic, otherwise print '**No**'.

Sample Input	Output for Sample Input
5 1 21 16161 523125 0	Case 1: Yes Case 2: No Case 3: Yes Case 4: No Case 5: Yes

1226 - One Unit Machine

OUM is a one unit machine which processes jobs. Since it can't handle heavyweight jobs; jobs needs to be partitioned into units. Initially, all the job information and unit partitions are given as input. Then the machine allocates necessary time slots. And in each time slot it asks the user for the name of the job to be processed. After getting the name; the machine determines the next unprocessed unit of that job and processes that unit in that slot. If there is no such unit, the machine crashes. A job is said to be complete if all the units of that job are complete.

For example, let J_1 and J_2 be two jobs each having 2 units. So, OUM will create 4 time slots. Now the user can give $J_1 J_2 J_2 J_1$ as input. That means it completes the 1st unit of J_1 in time slot 1 and then completes the 1st unit of J_2 in time slot 2. After that it completes the 2nd unit of J_2 and 2nd unit of J_1 in time slots 3 and 4 respectively. But if the user gives $J_1 J_1 J_2 J_1$ as input, the machine crashes in time slot 4 since it tries to process 3rd unit of J_1 which is not available.

Now, Sam is the owner of a software firm named ACM and he has n jobs to complete using OUM. He wants to complete Job_i before Job_{i+1} where $1 \leq i < n$. Now he wants to know the total number of ways he can complete these jobs without crashing the OUM. He assigned you for this task. Two ways are different if at t^{th} slot one processed a unit of Job_i and another processed a unit of Job_j , where $i \neq j$. For the example above, there are three ways:

$J_1 J_1 J_2 J_2$
 $J_1 J_2 J_1 J_2$
 $J_2 J_1 J_1 J_2$

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with an integer n ($1 \leq n \leq 1000$). The next line contains n space separated positive integers $k_1, k_2, k_3 \dots k_n$. Where, k_i denotes the number of units for the i^{th} job. You can assume that total number of units for all the jobs in any case is not greater than 10^6 .

Output

For each case, print the case number and the result modulo $1000,000,007$.

Sample Input	Output for Sample Input
2 2 2 2 3 2 2 3	Case 1: 3 Case 2: 45

1227 - Boiled Eggs

Three of the trouble-makers went to Malaysia this year. A rest house was booked for them. Unlike other rest houses, this rest house was like a normal duplex house. So, it had a kitchen. And the trouble-makers were given all the ingredients to cook, but they had to cook themselves.

None of them had any previous cooking experience, but they became very excited and planned to cook so many delicious foods! Ideas were coming from their minds like rains from clouds. So, they went to the super market and bought a lot of extra ingredients for their great recipes. For example, they bought **20** eggs. The excited trouble-makers returned to the rest house and found that the gas stove was not connected to the gas cylinder. So, they became very sad, because it was not possible for them to connect such complex thing. And so many foods were about to be rotten. But luckily, they found the microwave oven working. So, they tried to boil all the eggs using the microwave oven (may be, first time in history)! And they succeeded to boil the eggs!



Now they have **n** eggs and a bowl. They put some eggs in the bowl with some water. And after that they put the bowl into the oven to boil the eggs. It's risky to put more than **P** eggs in the bowl and the bowl can carry at most **Q** gm of eggs. It takes **12** minutes to boil a bowl of eggs. Now you are given the weight of the eggs in gm, and the trouble-makers have exactly **12** minutes in their hand. You have to find the maximum number of eggs they can boil without taking any risk.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with three integers **n** ($1 \leq n \leq 30$), **P** ($1 \leq P \leq 30$) and **Q** ($1 \leq Q \leq 30$). The next line contains **n** positive integers (not greater than **10**) in non-descending order. These integers denote the weight of the eggs in gm.

Output

For each case, print the case number and the desired result.

Sample Input	Output for Sample Input
2 3 2 10 1 2 3 4 5 5 4 4 5 5	Case 1: 2 Case 2: 1

1228 - e-Friends

Social networking web sites are very popular these days. I am not mentioning any names because you know better than me. People do have a lot of e-friends now, "you may not know his real identity but he is your friend". Sometimes, there can be rivalries between people, and thus people gets blocked or deleted. And these rivalries don't have to be necessarily commutative. That means **x** may think **y** as his enemy, but **y** may or may not think so.

However, what if you get the opportunity to see all your e-friends live? Since you like social networks very much, you planned to arrange a get-together with your e-friends. So, you invited **n** of your e-friends in your home. For simplicity, you numbered them from **1** to **n**. They were your friends, but rivalries existed amongst many of them. You asked them to form a queue such that they can get food from the table. And you will serve foods one by one. But your friends demanded that, no enemy should be the next person in the queue. That means if **x** thinks that **y** is his enemy and **x**'s position in the queue is **i**, then **y**'s position shouldn't be **i-1**.

So, it became very complex. That's why you decided to add a dissatisfaction index **k** if one's demand is not fulfilled. And the total dissatisfaction index is the summation of all the dissatisfaction indexes. Now you know the rivalries and the maximum total dissatisfaction index you may allow, you want to find the total number of possible arrangements.

Input

Input starts with an integer **T** (≤ 30), denoting the number of test cases.

Each case starts with three integers **n** ($1 \leq n \leq 12$), **k** ($0 \leq k \leq 10^6$) and **q** ($1 \leq q \leq 1000$) which denotes the number of queries. Then there will be **n** lines. The **ith** line contains an integer **t_i** ($0 \leq t_i < n$) and ids of **t_i** distinct friends. It means **ith** person thinks the **t_i** listed persons as his enemy. Each of the next **q** lines contains an integer **r** ($0 \leq r \leq 10^8$) denoting the total allowable dissatisfaction index.

Output

For each case, print the case number first. Then for each query, print the total number of arrangements possible not exceeding the given total dissatisfaction index.

Sample Input	Output for Sample Input
1 2 10 2 1 2 0 10 5	Case 1: 2 1

1229 – Treblecross

Treblecross is a two player game where the goal is to get three **X** in a row on a one-dimensional board. At the start of the game all cells in the board are empty. In each turn a player puts an **X** in an empty cell, and if the move results three **X** next to each other, that player wins.

Given the current state of the game, you are to determine if the current player to move can win the game assuming both players play optimally.

Consider the game where the board size is 5 cells. If the first player puts an **X** at position three (in the middle) so the state becomes ..**X**.., he will win the game as no matter where the other player puts his **X**, the first player can get three **X** in a row. If, on the other hand, the first player puts the **X** in any other position, the second player will win the game by putting the **X** in the opposite corner (for instance, after the second players move the state might be .**X**..**X**). This will force the first player to put an **X** in a position so the second player wins in the next move.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with a line containing a string denoting the current status of the game. The string will only contain the characters '**!**' and '**X**'. The length of the string (the size of the board) will be between **3** and **200** characters, inclusive. No state will contain three **X** in a row.

Output

For each case, print the case number and the positions on the board, where the player to move may put an **X** and win the game. The positions should be separated by a single space, and be in increasing order. The leftmost position on the board is **1**. If there is no such position print **0**.

Sample Input	Output for Sample Input
4X.....X..X.....X.....X...X .X.X....X	Case 1: 3 Case 2: 0 Case 3: 3 Case 4: 5 6 13 14

1230 – Placing Lampposts

As a part of the mission 'Beautification of Dhaka City', the government has decided to replace all the old lampposts with new expensive ones. Since the new ones are quite expensive and the budget is not up to the requirement, the government has decided to buy the minimum number of lampposts required to light the whole city.

Dhaka city can be modeled as an undirected graph with no cycles, multi-edges or loops. There are several roads and junctions. A lamppost can only be placed on junctions. These lampposts can emit light in all the directions and that means a lamppost that is placed in a junction will light all the roads leading away from it.

The 'Dhaka City Corporation' has given you the road map of Dhaka city. You are hired to find the minimum number of lampposts that will be required to light the whole city. These lampposts can then be placed on the required junctions to provide the service. There could be many combinations of placing these lampposts that will still cover all the roads. In that case, you have to place them in such a way so that the number of roads receiving light from two lampposts is maximized. (A careful thought will reveal that all the roads will get light either from one post or two posts).

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers **N ($1 \leq N \leq 1000$)** and **M ($0 \leq M < N$)** which indicate the number of junctions and roads respectively. The junctions are numbered from **0** to **N-1**. Each of the next **M** lines contains two integers **a** and **b ($0 \leq a, b < N, a \neq b$)**, which denotes that there is a road from junction **a** to **b**.

Output

For each case, print the case number, the minimum number of lampposts required to light the whole city, the number of roads that are receiving lights from two lampposts and the number of roads that are receiving light from only one lamppost.

Sample Input	Output for Sample Input
2 4 3 0 1 1 2 2 3 5 4 0 1 0 2 0 3 0 4	Case 1: 2 1 2 Case 2: 1 0 4

1231 – Coin Change (I)

In a strange shop there are n types of coins of value $A_1, A_2 \dots A_n$. $C_1, C_2, \dots C_n$ denote the number of coins of value $A_1, A_2 \dots A_n$ respectively. You have to find the number of ways you can make K using the coins.

For example, suppose there are three coins 1, 2, 5 and we can use coin 1 at most 3 times, coin 2 at most 2 times and coin 5 at most 1 time. Then if $K = 5$ the possible ways are:

```
1112  
122  
5
```

So, 5 can be made in 3 ways.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 50$) and K ($1 \leq K \leq 1000$). The next line contains $2n$ integers, denoting $A_1, A_2 \dots A_n, C_1, C_2 \dots C_n$ ($1 \leq A_i \leq 100, 1 \leq C_i \leq 20$). All A_i will be distinct.

Output

For each case, print the case number and the number of ways K can be made. Result can be large, so, print the result modulo **100000007**.

Sample Input	Output for Sample Input
2 3 5 1 2 5 3 2 1 4 20 1 2 3 4 8 4 2 1	Case 1: 3 Case 2: 9

1232 - Coin Change (II)

In a strange shop there are **n** types of coins of value $A_1, A_2 \dots A_n$. You have to find the number of ways you can make **K** using the coins. You can use any coin at most **K** times.

For example, suppose there are three coins 1, 2, 5. Then if **K = 5** the possible ways are:

11111
1112
122
5

So, 5 can be made in 4 ways.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers **n** ($1 \leq n \leq 100$) and **K** ($1 \leq K \leq 10000$). The next line contains **n** integers, denoting $A_1, A_2 \dots A_n$ ($1 \leq A_i \leq 500$). All A_i will be distinct.

Output

For each case, print the case number and the number of ways **K** can be made. Result can be large, so, print the result modulo **100000007**.

Sample Input	Output for Sample Input
2 3 5 1 2 5 4 20 1 2 3 4	Case 1: 4 Case 2: 108

1233 – Coin Change (III)

In a strange shop there are n types of coins of value $A_1, A_2 \dots A_n$. $C_1, C_2, \dots C_n$ denote the number of coins of value $A_1, A_2 \dots A_n$ respectively. You have to find the number of different values (from 1 to m), which can be produced using these coins.

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 100$), m ($0 \leq m \leq 10^5$). The next line contains $2n$ integers, denoting $A_1, A_2 \dots A_n, C_1, C_2 \dots C_n$ ($1 \leq A_i \leq 10^5, 1 \leq C_i \leq 1000$). All A_i will be distinct.

Output

For each case, print the case number and the result.

Sample Input	Output for Sample Input
2 3 10 1 2 4 2 1 1 2 5 1 4 2 1	Case 1: 8 Case 2: 4

1234 – Harmonic Number

In mathematics, the n^{th} harmonic number is the sum of the reciprocals of the first n natural numbers:

$$\begin{aligned} H_n &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} \\ &= \sum_{k=1}^n \frac{1}{k} \end{aligned}$$

In this problem, you are given n , you have to find H_n .

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 10^8$).

Output

For each case, print the case number and the n^{th} harmonic number. Errors less than 10^{-8} will be ignored.

Sample Input	Output for Sample Input
12	Case 1: 1
1	Case 2: 1.5
2	Case 3: 1.8333333333
3	Case 4: 2.0833333333
4	Case 5: 2.2833333333
5	Case 6: 2.450
6	Case 7: 2.5928571429
7	Case 8: 2.7178571429
8	Case 9: 2.8289682540
9	Case 10: 18.8925358988
90000000	Case 11: 18.9978964039
99999999	Case 12: 18.9978964139
100000000	

1235 – Coin Change (IV)

Given n coins, values of them are $A_1, A_2 \dots A_n$ respectively, you have to find whether you can pay K using the coins. You can use any coin at most two times.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 18$) and K ($1 \leq K \leq 10^9$). The next line contains n distinct integers denoting the values of the coins. These values will lie in the range $[1, 10^7]$.

Output

For each case, print the case number and 'Yes' if you can pay K using the coins, or 'No' if it's not possible.

Sample Input	Output for Sample Input
3 2 5 1 2 2 10 1 2 3 10 1 3 5	Case 1: Yes Case 2: No Case 3: Yes

1236 – Pairs Forming LCM

Find the result of the following code:

```
long long pairsFormLCM( int n ) {
    long long res = 0;
    for( int i = 1; i <= n; i++ )
        for( int j = i; j <= n; j++ )
            if( lcm(i, j) == n ) res++;
    return res;
}
```

A straight forward implementation of the code may time out. If you analyze the code, you will find that the code actually counts the number of pairs (i, j) for which $\text{lcm}(i, j) = n$ and $(i \leq j)$.

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 10^{14}$).

Output

For each case, print the case number and the value returned by the function '`pairsFormLCM(n)`'.

Sample Input	Output for Sample Input
15	Case 1: 2
2	Case 2: 2
3	Case 3: 3
4	Case 4: 5
6	Case 5: 4
8	Case 6: 5
10	Case 7: 8
12	Case 8: 5
15	Case 9: 8
18	Case 10: 8
20	Case 11: 5
21	Case 12: 11
24	Case 13: 3
25	Case 14: 4
27	Case 15: 2
29	

1237 – Cyber Café

After graduating in CS, Mr 'AcmHateKori' has opened a cyber café. His business was running quite well unless one day he faced a problem, and with his non-ACM mind he was unable to solve that.

In his café he maintains two papers for his customers. One paper contains the entering times of the customers in the café and the other paper contains the times they have exited. He writes the times in the same order in the papers for the money calculations. And he only writes the nearest minute of the time. He writes the times from **0** to **1000** (0 means 12:00 am, 60 means 01:00 am, 730 means 02:10 pm). This idea was given by his little son.

When a person exits the café he was to pay some money. If he stays in the café for **T** minutes, he has to pay $(T-K)^2$ paisa but not more than **G** paisa (if money exceeds **G**). It's guaranteed that every person stays at least one minute in the café.

Now one day his little son came to the café and took the paper that contained the entering times of the customers. He took a new paper and wrote the entering times randomly and threw the old paper. When his father came, he found two papers, where the first one contained some random entering times. He was at a loss and angry. Cause how can he find the total money given by the customers since for an entering time there can be multiple exiting times.

So, finally he realized the importance of ACM in life and asked you to find the minimum and maximum money he could earn by matching all entering times and exiting times. His son can be faulty. So, you have to report that too.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing three integers **n** ($1 \leq n \leq 50$), **K** ($1 \leq K \leq 1000$), **G** ($0 \leq G \leq 10000$) where **n** denotes the number of customers. The next line contains **n** space separated integers denoting the entering times of the customers. The next line contains **n** space separated integers denoting the exiting times of the customers. All the given times will lie in the range [0, 1000].

Output

For each case, print the case number, the minimum and maximum money he could earn or 'impossible' if his son had made some mistakes.

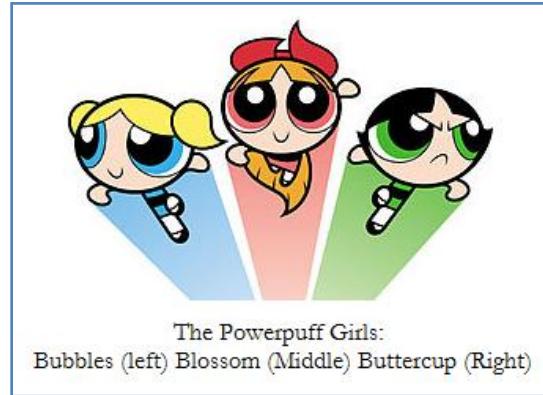
Sample Input	Output for Sample Input
2 3 7 11 8 9 10 9 11 20 2 10 10 1 11 2 9	Case 1: 31 33 Case 2: impossible

1238 - Power Puff Girls

The city of Townsville! This nice city is the home for the power puff girls - Blossom, Bubbles and Buttercup. To introduce their personality we can sing a song:

Blossom, commander and the leader;
Bubbles, she is the joy and the laughter;
Buttercup, she is the toughest fighter.

These super girls defend Townsville from monsters and super villains using their super powers, intelligence. They live in a home in Townsville with the professor who is like their father.



The girls are young now and they don't like to fight the monsters anymore. So, if they are outside their home, they are found shopping. And when they get back to home, they simply watch the TV serials. It's such a horrible fact that the super intelligent girls are wasting their time watching serials that consist of the rivalries between Wives and their Mother in Laws. And when they use computers they are usually found using the facenote (a dangerous¹ social networking site).

So, such wonderful girls just became lazy and useless. Often they are seen fighting each other, the comments that can be heard, are like, 'Tulsi is the best.', 'Gopi is better than Rashee.' The professor is quite upset with the girls, and he can't even watch any science show or sports because of these irritating serials.

So, the professor made a plan and asked the girls to go for shopping such that he can watch an important science show in the TV. The girls became very excited and they went out for shopping. But soon they realize that one of their favorite serials will start soon, and they need to get back home for that serial.

To be more specific let's consider the city as a 2D grid. In the grid there are some symbols, the meaning of them are:

1. '.' means an empty place
2. 'a' denotes the position of Blossom
3. 'b' denotes the position of Bubbles
4. 'c' denotes the position of Buttercup
5. 'm' denotes that there is a monster
6. 'h' denotes their home
7. '#' denotes a wall and the girls can't pass through it

The three girls move simultaneously. And in each minute, from their current cells, they can move to any four adjacent cells (North, East, West, South) if the destination cell is neither a wall nor the cell

contains a monster. Because they want to get home as soon as possible, they want to avoid the monsters. You can assume that they can move to a common cell if necessary.

Now you are given the information of the city and their positions. You have to find the minimum time when they all are in home.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers: **m** and **n ($4 \leq m, n \leq 20$)**, **m** denotes the number of rows and **n** denotes the number of columns of the modeled grid. Each of the next **m** lines contains **n** characters representing the city.

You can assume that 'a', 'b', 'c', 'h' will occur exactly once in the grid. The outer boundaries will always be marked by '#'.

Output

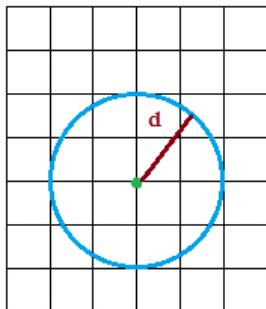
For each case, print the case number and the minimum time needed when they all are in their home. You can assume that a solution will always exist.

Sample Input	Output for Sample Input
2 6 8 ##### #...c..# #.....# #.a.h.b# #.....# ##### 5 9 ##### #mmm...c# #ma.h## #m....b.# #####	Case 1: 2 Case 2: 4

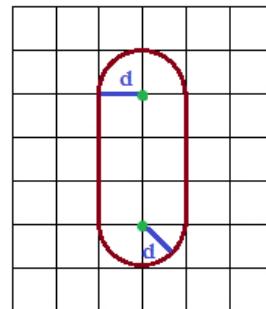
1239 – Convex Fence

I have a land consisting of **n** trees. Since the trees are favorites to cows, I have a big problem saving them. So, I have planned to make a fence around the trees. I want the fence to be convex (curves are allowed) and the minimum distance from any tree to the fence is at least **d** units. And definitely I want a single big fence that covers all trees.

You are given all the information of the trees, to be specific, the land is shown as a 2D plane and the trees are plotted as 2D points. You have to find the perimeter of the fence that I need to create as described above. And you have to minimize the perimeter.



Only one tree, so a circular fence is needed



Two trees, the fence is shown

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with a line containing two integers **n** ($1 \leq n \leq 50000$), **d** ($1 \leq d \leq 1000$). Each of the next lines contains two integers **x_i y_i** ($-10^8 \leq x_i, y_i \leq 10^8$) denoting a position of a tree. You can assume that all the positions are distinct.

Output

For each case, print the case number and the minimum possible perimeter of the fence. Errors less than 10^{-3} will be ignored.

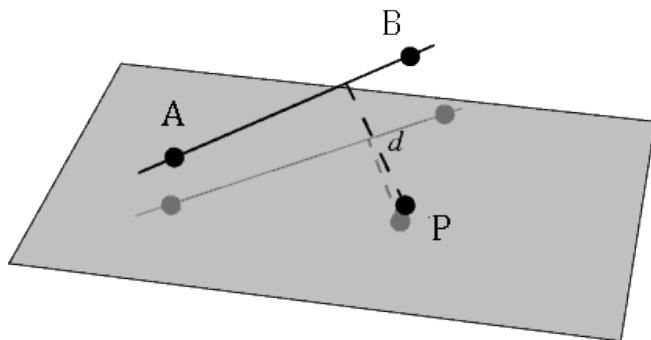
Sample Input	Output for Sample Input
3 1 2 0 0 2 1 0 -1 0 2 3 5 0 0 5 0 0 5	Case 1: 12.566370614 Case 2: 12.2831853 Case 3: 48.4869943478

Note

Dataset is huge, use faster i/o methods.

1240 – Point Segment Distance (3D)

Given a segment in 3D space, identified by $\mathbf{A}(x_1, y_1, z_1)$, $\mathbf{B}(x_2, y_2, z_2)$ and another point $\mathbf{P}(x, y, z)$ your task is to find the minimum possible Euclidean distance between the point \mathbf{P} and the segment \mathbf{AB} .



Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case starts with a line containing nine integers $x_1, y_1, z_1, x_2, y_2, z_2, x, y, z$. The magnitude of any integer will not be greater than **100**.

Output

For each case, print the case number and the distance. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 0 0 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 1	Case 1: 1 Case 2: 0.8164965809

1241 - Pinocchio

You must have heard the name of Pinocchio. If you never heard of him, don't panic, I am here to help you. But why I am introducing you to Pinocchio? Cause there is an interesting (and also quite strange) fact about him.

Pinocchio is a boy who lives in a certain village. He is a little boy, who is prone to telling lies, fabricating stories and exaggerating or creating tall tales for various reasons. But the strange fact is, when he does this, his nose gets longer. But when he tells the truth his nose gets back to normal size which is 2 cm.

Usually, when he wakes up in the morning his nose gets back to normal size. When he tells a lie, his nose grows at least 1 cm and at most 5 cm.



Pinocchio



Pinocchio after some lies

There is a common paradox related to him. What if he says, "**My nose grows now.**" You may wonder why the simple looking statement leads to a paradox. The result of this deadly statement is noted below:

Assume that this sentence is true

- Which means that Pinocchio's nose grows now because he truthfully says it is, but then
- Pinocchio's nose does not grow now because it grows only as Pinocchio lies, but then
- Pinocchio's nose grows now because Pinocchio's nose does not grow now, and Pinocchio trustfully says it grows now, and it is false, that makes Pinocchio's sentence to be false, but then
- Pinocchio's nose does not grow now because Pinocchio's nose grows now, and Pinocchio trustfully says it grows now, and it is true that makes Pinocchio's sentence to be true, but then
- And so on ad infinitum.

Now assume that the sentence is false

- Which means that Pinocchio's nose does not grow now because he falsely says it is, but then
- Pinocchio's nose grows now because it grows only as Pinocchio lies, but then
- Pinocchio's nose does not grow now because Pinocchio's nose grows now, and Pinocchio falsely says it grows now, and it is false that makes Pinocchio's sentence to be true, but then
- Pinocchio's nose grows now because Pinocchio's nose does not grow now, and Pinocchio falsely says it grows now, and it is true, that makes Pinocchio's sentence to be false, but then
- And so on ad infinitum.

Now you are given some sizes of his nose in a day. Assume that he hasn't told any truth in that day and the sizes are reported in increasing order of time. You have to find the minimum number of lies he has told in that day such that the report of the sizes is true.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 10$)**, where **n** denotes the total number of reports of his nose in a certain day. The next line contains **n** space separated integers denoting the sizes of his nose in that day. If the integers in that line is **a₁, a₂ ... a_n**, you can assume that

$$(2 \leq a_1 \leq a_2 \leq \dots \leq a_n \leq 50)$$

Output

For each case, print the case number and the minimum number of lies Pinocchio has told in that day. See the samples for the output format.

Sample Input	Output for Sample Input
<pre>2 5 2 3 3 3 4 4 2 3 4 5</pre>	<pre>Case 1: 2 Case 2: 3</pre>

1242 – Similar Trees (II)

Tom and Jerry like to draw trees. One day Tom drew a tree, and soon he found another tree drawn by Jerry. He thinks that Jerry has copied his tree and added some nodes and edges, and for safety he may have changed the label of the nodes (but not the root). So, Tom asks your help.

A tree is a connected graph with no cycles. They both drew rooted trees, now your task is to find whether Jerry has modified Tom's tree or not.

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a line containing an integer n ($2 \leq n \leq 100$) denoting the number of nodes in Jerry's tree. Each of the next $n-1$ lines contains two integers $u_i v_i$ ($1 \leq u_i, v_i \leq n, u_i \neq v_i$) denoting that there is an edge between u_i and v_i in Jerry's tree.

The next line contains an integer m ($1 \leq m < n$) denoting the number of nodes in Tom's tree. Each of the next $m-1$ lines contains two integers $p_i q_i$ ($1 \leq p_i, q_i \leq m, p_i \neq q_i$) denoting that there is an edge between p_i and q_i in Tom's tree.

For both trees, **1** is the root.

Output

For each case, print the case number and '**Yes**' if Jerry has modified Tom's tree or '**No**' otherwise.

Sample Input	Output for Sample Input
2 4 1 2 2 4 1 3 3 1 3 3 2 5 1 2 2 4 1 3 3 5 4 1 3 1 2 1 4	Case 1: Yes Case 2: No

1243 - Guardian Knights

The kingdom of Azkaroth is in great danger. Some strange creatures have captured the city and attacking the poor inhabitants.

The old wise king has chosen his best knights to save the kingdom from these creatures. They have already found that the creatures are targeting some food mills. So, the best way is to send the knights to look after the food mills. And if anything goes wrong he may call others to attack the creatures.

The kingdom can be modeled as an $n \times n$ grid. The legends are:

- '#' denotes a rock
- '.' denotes an empty space
- [A-Z] denotes a knight
- 'm' denotes a mill

A knight can go to a mill using some moves, in each move a knight can go to its **four** neighboring cells {North, East, West, South} with the restriction that the destination cell shouldn't contain a rock (multiple knights can move together in a cell). And the number of mills that can be maintained by a knight is limited. The cost to look after a mill by a knight is the distance between the knight and the mill, distance means the number of moves the knight has to use to go to the mill. If a knight looks after multiple mills, the cost is the summation of distances from the knight to the mills.

Now your task is to find the minimum total cost needed for the knights to look after all the mills. There can be more than one knight looking after a mill.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing three integers **n ($5 \leq n \leq 30$)**, **k ($1 \leq k \leq 26$)** and **m ($1 \leq m \leq 100$)** where **k** denotes the number of knights and **m** denotes the number of mills in the grid. The knights will be marked by first **k** capital letters and the grid contains exactly one letter of each kind. Each of the next **n** lines contains **n** characters denoting the grid. The next line contains **k** space separated integers (between **1** and **100**) denoting the number of mills the **ith** knight can look after. That means the first integer denotes the number of mills that can be looked after by knight '**A**', second integer denotes the number of mills looked after by knight '**B**' and so on. And the outer cells of the grid will be rocks.

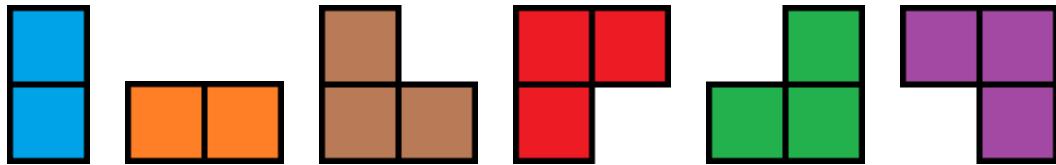
Output

For each case, print the case number and the minimum cost as described above. You can assume that a solution will always exist.

Sample Input	Output for Sample Input
2 7 4 5 ##### #A..mD# #. . . m# #. . m.m# #. . . m# #B...C# ##### 1 2 1 1 7 3 6 ##### #A#.m.# #. # ..m# #m#m.m# #. # . . m# #B...C# ##### 1 2 3	Case 1: 15 Case 2: 19

1244 - Tiles

There is a huge $2 \times N$ board, six types of tiles are available, and each of them is infinitely many, you have to find the number of ways you can fill the board using the tiles. Two board configurations are different if at least in one cell, their colors differ. The tiles are given below:



You **cannot** rotate or flip any tile. And no cell in the board should be empty. The tiles shouldn't overlap.

For example, a 2×3 board can be colored by 5 ways, they are:



Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case starts with a line containing an integer N ($1 \leq N \leq 10^9$).

Output

For each case, print the case number and the number of ways the board can be colored. The number may be large, so, output the number modulo **10007**.

Sample Input	Output for Sample Input
5	Case 1: 5
3	Case 2: 1255
10	Case 3: 6239
20	Case 4: 6542
100	Case 5: 5502
87	

1245 – Harmonic Number (II)

I was trying to solve problem '1234 – Harmonic Number', I wrote the following code

```
long long H( int n ) {
    long long res = 0;
    for( int i = 1; i <= n; i++ )
        res = res + n / i;
    return res;
}
```

Yes, my error was that I was using the integer divisions only. However, you are given **n**, you have to find **H(n)** as in my code.

Input

Input starts with an integer **T (≤ 1000)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n < 2^{31}$)**.

Output

For each case, print the case number and **H(n)** calculated by the code.

Sample Input	Output for Sample Input
11	Case 1: 1
1	Case 2: 3
2	Case 3: 5
3	Case 4: 8
4	Case 5: 10
5	Case 6: 14
6	Case 7: 16
7	Case 8: 20
8	Case 9: 23
9	Case 10: 27
10	Case 11: 46475828386
2147483647	

1246 - Colorful Board

You are given a rectangular board. You are asked to draw **M** horizontal lines and **N** vertical lines in that board, so that the whole board will be divided into **(M+1) x (N+1)** cells. So, there will be **M+1** rows each of which will exactly contain **N+1** cells or columns. The **yth** cell of **xth** row can be called as **cell(x, y)**. The distance between two cells is the summation of row difference and column difference of those two cells. So, the distance between **cell(x₁, y₁)** and **cell(x₂, y₂)** is

$$|x_1 - x_2| + |y_1 - y_2|$$

For example, the distance between cell (2, 3) and cell (3, 2) is $|2 - 3| + |3 - 2| = 1 + 1 = 2$.

After that you have to color every cell of the board. For that you are given **K** different colors. To make the board more beautiful you have to make sure that no two cells having the same color can have odd distance between them. For example, if you color cell (3, 5) with red, you cannot color cell (5, 8) with red, as the distance between them is 5, which is odd. Note that you can keep some color unused, but you can't keep some cell uncolored.

You have to determine how many ways to color the board using those **K** colors.

Input

Input starts with an integer **T (≤ 20000)**, denoting the number of test cases.

Each case starts with a line containing three integers **M, N, K ($0 \leq M, N \leq 19, 1 \leq K \leq 50$)**.

Output

For each case, print the case number and the number of ways you can color the board. The result can be large, so print the result modulo **1000000007**.

Sample Input	Output for Sample Input
4 0 0 1 0 0 2 5 5 2 5 5 1	Case 1: 1 Case 2: 2 Case 3: 2 Case 4: 0

1247 - Matrix Game

Given an $m \times n$ matrix, where m denotes the number of rows and n denotes the number of columns and in each cell a pile of stones is given. For example, let there be a 2×3 matrix, and the piles are

2 3 8
5 2 7

That means that in cell(1, 1) there is a pile with 2 stones, in cell(1, 2) there is a pile with 3 stones and so on.

Now Alice and Bob are playing a strange game in this matrix. Alice starts first and they alternate turns. In each turn a player selects a row, and can draw any number of stones from any number of cells in that row. But he/she must draw at least one stone. For example, if Alice chooses the 2nd row in the given matrix, she can pick 2 stones from cell(2, 1), 0 stones from cell (2, 2), 7 stones from cell(2, 3). Or she can pick 5 stones from cell(2, 1), 1 stone from cell(2, 2), 4 stones from cell(2, 3). There are many other ways but she must pick at least one stone from all piles. The player who can't take any stones loses.

Now if both play optimally who will win?

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers: m and n ($1 \leq m, n \leq 50$). Each of the next m lines contains n space separated integers that form the matrix. All the integers will be between 0 and 10^9 (inclusive).

Output

For each case, print the case number and '**Alice**' if Alice wins, or '**Bob**' otherwise.

Sample Input	Output for Sample Input
2 2 3 2 3 8 5 2 7 2 3 1 2 3 3 2 1	Case 1: Alice Case 2: Bob

1248 – Dice (III)

Given a dice with **n** sides, you have to find the expected number of times you have to throw that dice to see all its faces at least once. Assume that the dice is fair, that means when you throw the dice, the probability of occurring any face is equal.

For example, for a fair two sided coin, the result is 3. Because when you first throw the coin, you will definitely see a new face. If you throw the coin again, the chance of getting the opposite side is 0.5, and the chance of getting the same side is 0.5. So, the result is

$$\begin{aligned} & 1 + (1 + 0.5 * (1 + 0.5 * ...)) \\ &= 2 + 0.5 + 0.5^2 + 0.5^3 + \dots \\ &= 2 + 1 = 3 \end{aligned}$$

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 10^5$)**.

Output

For each case, print the case number and the expected number of times you have to throw the dice to see all its faces at least once. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
5 1 2 3 6 100	Case 1: 1 Case 2: 3 Case 3: 5.5 Case 4: 14.7 Case 5: 518.7377517640

1249 – Chocolate Thief

I gave some chocolates to students for their extraordinary performances. A chocolate is a cube shaped thing, which has length, width and height. All the students got the same amount of chocolates; their dimensions may be different but the volumes are same.

Now some of the students are claiming that there is one chocolate thief amongst them. So, it's not an easy task for me to find the chocolate thief, so I am asking your help.

You are given the names of the students and the dimensions of their chocolates; you have to find the name of the chocolate thief. You can assume that there can be at most one thief and if there is a thief, he took some portion of the chocolate from another student (not students).

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($2 \leq n \leq 100$)** denoting the number of students. Each of the next **n** lines contains a name and three integers denoting the length, width and height of his current chocolate share. Names are strings containing alphanumeric characters and the length of a name is between **1** and **20**. And length, width and height will lie in the range **[1, 100]**. Input follows the above restrictions.

Output

For each case, print the case number first. Then if no thief is found, print '**no thief**'. Otherwise print '**x took chocolate from y**' where **x** is the name of the chocolate thief, and **y** is the name of the person from whom the chocolate was taken.

Sample Input	Output for Sample Input
2 11 atq 3 4 3 mun 10 4 1 sam1 6 6 1 sam2 18 2 1 mub 1 36 1 tan 1 4 9 sha 4 3 3 di 3 12 1 nab 2 2 9 all 8 4 1 fah 3 2 6 2 ja 10 10 10 em 2 50 10	Case 1: mun took chocolate from all Case 2: no thief

1250 - Village Postman

I postman has to deliver post to villagers who live in several villages and in the roads that connect the villages. Actually there are houses in villages as well as along the sides of the roads. There are **n** villages and **m** roads. Villages are numbered from **1** to **n**, and all the villages are connected by some paths. A path consists of some connected roads. The postman starts his journey from village 1 and he must finish his journey at village 1. So, he wants to find a tour that visits each village, each road at least once. Since all the villagers want the postman to come in their house as early as possible; they designed a cost service.

If an unvisited village **i** is visited as the **kth** (1 indexed) **different** village on the tour and **k ≤ w(i)**, the village pays **w(i) - k** taka to the post. However, if **k > w(i)**, the post agrees to pay **k - w(i)** taka to the village. Moreover, the post has to spend one taka for each road the postman visits. Even if the postman travels a road **p** times, then the post has to spend **p** taka. The villages are established in a way that such a tour is always possible. And there are always 2, 4 or 8 roads going out from each village. Now your task is to help the post to find a route that maximizes the earning (or minimizes the loss).

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers **n (1 ≤ n ≤ 200)** and **m**. The next line contains **n** space separated integers. The **ith** integer denotes the **w(i)** for the **ith** village (**0 ≤ w(i) ≤ 1000**). Each of the next **m** lines contains two integers **u v (1 ≤ u, v ≤ n)** denoting that there is a road between village **u** and **v**. You can assume that the given input follows the constraints given above.

Output

For each case, print the case number and the maximum profit. Then in the next line print the route. So, this line should contain numbers of consecutive villages on the route **v₁ v₂ ... v_x** separated by single spaces, with **v₁ = v_x = 1**. There can be several solutions, any valid one will do.

Sample Input	Output for Sample Input
1 6 7 0 7 4 10 20 5 2 4 1 5 2 1 4 5 3 6 1 6 1 3	Case 1: 18 1 5 4 2 1 6 3 1

Note

This is a special judge problem, wrong output format may cause '**wrong answer**'.

1251 – Forming the Council

In a city there are n voters, and m people formed the Govt. council. The council members are numbered from 1 to m . Now everyone is complaining that the council is biased. So, they made a plan. The plan is that the voters are given a chance to vote again to form the new council. A vote will be like $\pm i \pm j$. '+' means the voter wants that member to be in the council, '-' means the voter doesn't want the member to be in the council. For example, there are 4 voters, they voted like

- +1 -3 the voter wants member 1 to be kept in the council or member 3 to be thrown out
- +2 +3 the voter wants member 2 to be kept in the council or member 3 to be kept in the council
- 1 -2 the voter wants member 1 to be thrown out or member 2 to be thrown out
- 4 +1 the voter wants member 4 to be thrown out or member 1 to be kept in the council

A voter will be satisfied if at least one of his wishes becomes true. Now your task is to form the council such that all the voters are happy.

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 20000$) and m ($1 \leq m \leq 8000$). Each of the next n lines contains a vote in the form $\pm i \pm j$ ($1 \leq i, j \leq m$).

Output

For each case, print the case number and 'Yes' if a solution exists, or 'No' if there is no solution. Then if the result is yes, print another line containing the number of members in the council followed by the members in ascending order. And print a single space between two numbers. There can be many solutions. Any valid one will do.

Sample Input	Output for Sample Input
3 4 3 +1 +3 +2 -1 +2 -3 -1 -2 4 2 +1 -2 +1 +2 -1 -2 -1 +2 1 3 +1 -3	Case 1: Yes 2 2 3 Case 2: No Case 3: Yes 0

Note

This is a special judge problem. Wrong output format may cause wrong answer.

1252 - Maintaining Communities

Tracking Communities in social networks, like facebook, twitter etc, are one of the most exiting works in the field of Artificial Intelligence now-a-days. A community is a group of people who are connected. Two persons are connected and form a community if they are directly connected or connected via some other persons. That means if person **A** and **B** are connected and **B** and **C** are also connected then **A** and **C** are connected and they all belong to the same community. It is quite obvious that, one single person can also form a community by himself.

Now you are given a description of a community where there are **n** persons and the connections between some pairs of the persons. You can assume that they do form a community and their connection is given such that if you want to find two peoples connectivity information, you will find exactly one path between them.

Now maintaining a connection between a pair requires some cost. And unfortunately, the social networking site cannot afford keeping a community which requires a cost of more than **K**. For example, say A and B and B and C are connected and $\text{cost}(A, B) = 5$, $\text{cost}(B, C) = 2$. Then if $K \geq 7$, they can afford this community. Otherwise they cannot.

So, they have made a plan, and that is they will break the community. They want to break the community into multiple communities such that each community requires maintenance cost no more than **K**. But if there are too many communities people may leave the network, that's why they want the minimum number of communities. Since you are the best, they find no option but to ask you.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with a line containing two integers **n** ($1 \leq n \leq 100$) and **K** ($1 \leq K \leq 100$). Each of the next **n - 1** lines contains three integers **u v w** ($1 \leq u, v \leq n$, $1 \leq w \leq 100$, $u \neq v$) meaning that there is a connection between person **u** and **v** and the maintenance cost of this connection is **w**.

Output

For each case, print the case number and the minimum number of communities they have to maintain.

Sample Input	Output for Sample Input
2 3 1 1 2 2 2 3 2 4 12 1 2 5 2 3 10 1 4 7	Case 1: 3 Case 2: 2

1253 - Misère Nim

Alice and Bob are playing game of **Misère Nim**. Misère Nim is a game playing on k piles of stones, each pile containing one or more stones. The players alternate turns and in each turn a player can select one of the piles and can remove as many stones from that pile unless the pile is empty. In each turn a player must remove at least one stone from any pile. Alice starts first. The player who removes the last stone **loses** the game.

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case starts with a line containing an integer k ($1 \leq k \leq 100$). The next line contains k space separated integers denoting the number of stones in each pile. The number of stones in a pile lies in the range $[1, 10^9]$.

Output

For each case, print the case number and '**Alice**' if Alice wins otherwise print '**Bob**'.

Sample Input	Output for Sample Input
3 4 2 3 4 5 5 1 1 2 4 10 1 1	Case 1: Bob Case 2: Alice Case 3: Bob

1254 - Prison Break

Michael Scofield has just broken out of the prison. Now he wants to go to a certain city for his next unfinished job. As you are the only programmer on his gang, he asked your help. As you know that the fuel prices vary in the cities, you have to write a code to help Scofield that instructs him where to take the fuel and which path to choose. Assume that his car uses one unit of fuel in one unit of distance. Now he gives you the starting city **s** where he starts his journey with his car, the destination city **t** and the capacity of the fuel tank of his car **c**, the code should find the route that uses the cheapest fuel cost. You can assume that Scofield's car starts with an empty fuel tank.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

Each case starts with a line containing two integers **n** ($2 \leq n \leq 100$) and **m** ($0 \leq m \leq 1000$) where **n** denotes the number of cities and **m** denotes the number of roads. The next line contains **n** space separated integers, each lies between 1 and 100. The **ith** integer in this line denotes the fuel price (per unit) in the **ith** city. Each of the next **m** lines contains three integers **u v w** ($0 \leq u, v < n, 1 \leq w \leq 100, u \neq v$) denoting that there is a road between city **u** and **v** whose length is **w**.

The next line contains an integer **q** ($1 \leq q \leq 100$) denoting the number of queries by Scofield. Each of the next **q** lines contains the request. Each request contains three integers: **c s t** ($1 \leq c \leq 100, 0 \leq s, t < n$) where **c** denotes the capacity of the tank, **s** denotes the starting city and **t** denotes the destination city.

Output

For each case, print the case number first. Then for each query print the cheapest trip from **s** to **t** using the car with the given capacity **c** or '**impossible**' if there is no way of getting from **s** to **t** with the given car.

Sample Input	Output for Sample Input
1 5 5 10 10 20 12 13 0 1 9 0 2 8 1 2 1 1 3 11 2 3 7 2 10 0 3 20 1 4	Case 1: 170 impossible

1255 – Substring Frequency

A string is a finite sequence of symbols that are chosen from an alphabet. In this problem you are given two non-empty strings **A** and **B**, both contain lower case English alphabets. You have to find the number of times **B** occurs as a substring of **A**.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

Each case starts with two lines. First line contains **A** and second line contains **B**. You can assume than $1 \leq \text{length}(A), \text{length}(B) \leq 10^6$.

Output

For each case, print the case number and the number of times **B** occurs as a substring of **A**.

Sample Input	Output for Sample Input
4 axbyczd abc abcabcaabcabc abc aabacbaabbaaz aab aaaaaa aa	Case 1: 0 Case 2: 4 Case 3: 2 Case 4: 5

Note

Dataset is huge, use faster I/O methods.

1256 - Word Puzzle

You are given a puzzle board where there are **n** words. Initially they are scattered. Your task is to find a order of the words such that the first character of **ith** word is same as the last character of the **(i-1)th** word ($1 < i \leq n$).

For example, you are given {"abef", "pqrs", "fzzp", "zama", "pxrp"}, the solution is

zama abcf fzzp pxrp pqrs

Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

Each case starts with a line containing two an integer **n** ($1 \leq n \leq 1000$). Each of the next **n** lines contains a word whose length is between **1** and **20** (inclusive). You can assume that the words contain lowercase letters only.

Output

For each case, print the case number and '**Yes**' if such a ordering can be possible, or '**No**' if there is no such ordering. If the result is yes, then in the next line you should print the **n** words in a valid order. Print a single space between two consecutive words. There can be multiple solutions, any valid one will do.

Sample Input	Output for Sample Input
2 5 abcf pqrs fzzp zama pxrp 3 yes no notsolvable	Case 1: Yes zama abcf fzzp pxrp pqrs Case 2: No

Note

This is a special judge problem. Wrong output format may cause wrong answer.

1257 – Farthest Nodes in a Tree (II)

Given a tree (a connected graph with no cycles), you have to find the cost to go to the farthest node from each node. The edges of the tree are weighted and undirected.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with an integer **n** ($2 \leq n \leq 30000$) denoting the total number of nodes in the tree. The nodes are numbered from **0** to **n-1**. Each of the next **n-1** lines will contain three integers **u v w** ($0 \leq u, v < n, u \neq v, 1 \leq w \leq 10000$) denoting that node **u** and **v** are connected by an edge whose weight is **w**. You can assume that the input will form a valid tree.

Output

For each case, print the case number in a line first. Then for each node (from **0** to **n - 1**) print the cost to go to the farthest node in a separate line.

Sample Input	Output for Sample Input
2 4 0 1 20 1 2 30 2 3 50 5 0 2 20 2 1 10 0 3 29 0 4 50	Case 1: 100 80 50 100 Case 2: 50 80 70 79 80

Note

Dataset is huge, use faster I/O methods.

1258 - Making Huge Palindromes

A string is said to be a palindrome if it remains same when read backwards. So, 'abba', 'madam' both are palindromes, but 'adam' is not.

Now you are given a non-empty string **S**, containing only lowercase English letters. The given string may or may not be palindrome. Your task is to make it a palindrome. But you are only allowed to add characters at the right side of the string. And of course you can add any character you want, but the resulting string has to be a palindrome, and the length of the palindrome should be as small as possible.

For example, the string is '**bababa**'. You can make many palindromes including

bababababab

babababab

bababab

Since we want a palindrome with minimum length, the solution is '**bababab**' cause its length is minimum.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with a line containing a string **S**. You can assume that $1 \leq \text{length}(S) \leq 10^6$.

Output

For each case, print the case number and the length of the shortest palindrome you can make with **S**.

Sample Input	Output for Sample Input
4 bababababa pqrs madamimadam anncbaaababaaa	Case 1: 11 Case 2: 7 Case 3: 11 Case 4: 19

Note

Dataset is huge, use faster I/O methods.

1259 – Goldbach's Conjecture

Goldbach's conjecture is one of the oldest unsolved problems in number theory and in all of mathematics. It states:

Every even integer, greater than 2, can be expressed as the sum of two primes [1].

Now your task is to check whether this conjecture holds for integers up to 10^7 .

Input

Input starts with an integer **T (≤ 300)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($4 \leq n \leq 10^7$, n is even)**.

Output

For each case, print the case number and the number of ways you can express **n** as sum of two primes. To be more specific, we want to find the number of **(a, b)** where

- 1) Both **a** and **b** are prime
- 2) **a + b = n**
- 3) **a ≤ b**

Sample Input	Output for Sample Input
2 6 4	Case 1: 1 Case 2: 1

Note

1. An integer is said to be prime, if it is divisible by exactly two different integers. First few primes are 2, 3, 5, 7, 11, 13, ...

1260 - Race Track

Once there was a country named Ajobdesh. One of the architects designed a race track in a famous stadium in that country.

The stadium was like a polygon. The architect made another polygon inside that stadium and the space in between the two polygons was the track. Unlike other countries their cars were circular.

Now you are asked to find the radius of the largest car that can complete the track. Or you can say that the car can move freely around the track.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($3 \leq n \leq 100$)** denoting the number of vertices of the polygon that was made by the architect. Each of the next **n** lines contains two integers **x_i y_i** denoting the co-ordinates of that polygon.

The next line contains an integer **m ($3 \leq m \leq 100$)** denoting the number of vertices of the polygon that formed the stadium. Each of the next **m** lines contains two integers **x_i y_i** denoting the co-ordinates of that polygon.

All coordinates have absolute value no larger than **1000**. The points of the polygons can be given in either clockwise or counterclockwise order and the two polygons do not intersect or touch themselves or each other. The outer polygon encloses the inner polygon.

Output

For each case, print the case number and the radius of the largest car that can complete the track. Errors less than **10^{-6}** will be ignored.

Sample Input	Output for Sample Input
1 3 0 0 1 0 1 1 5 3 -3 3 3 -4 2 -1 -1 -2 -2	Case 1: 0.7071067812

1261 – K-SAT Problem

SAT was the first known NP-complete problem. The problem remains NP-complete even if all expressions are written in conjunctive normal form with 3 variables per clause (3-CNF), yielding the 3-SAT problem. A **K-SAT** problem can be described as follows:

There are **n** persons, and **m** objects. Each person makes **K** wishes, for each of these wishes either he wants to take an object or he wants to reject an object. You have to take a subset of the objects such that every person is happy. A person is happy if at least **one** of his **K** wishes is kept. For example, there are 3 persons, 4 objects, and **K = 2**, and

Person 1 says, "take object 1 or reject 2."

Person 2 says, "take object 3 or 4."

Person 3 says, "reject object 3 or 1."

So, if we take object 1 2 3, then it is not a valid solution, since person 3 becomes unhappy. But if we take 1 2 4 then everyone becomes happy. If we take only 4, it's also a valid solution. Now you are given the information about the persons' wishes and the solution we are currently thinking. You have to say whether the solution is correct or not.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing three integers **n, m, K ($1 \leq n, m, K \leq 30$)**. Each of the next **n** lines contains **K** space separated integers where the **ith** line denotes the wishes of the **ith** person. Each of the integers in a line will be either positive or negative. Positive means the person wants the object in the solution; negative means the person doesn't want that in the solution. You can assume that the absolute value of each of the integers will lie between **1** and **m**.

The next line contains an integer **p ($0 \leq p \leq m$)** denoting the number of integers in the solution, followed by **p** space separated integers each between **1** and **m**, denoting the solution. That means the objects we have taken as solution set.

Output

For each case, print the case number and '**Yes**' if the solution is valid or '**No**' otherwise.

Sample Input	Output for Sample Input
2 3 4 2 +1 -2 +3 +4 -3 -1 1 4 1 5 3 +1 -2 +4 2 2 5	Case 1: Yes Case 2: No

1262 – Diagonal Sum

You are given an $m \times n$ grid containing an integer in each cell. You have a software that sends rays thorough each diagonal and calculates the sum of all the diagonals. The software sends the rays in two phases, in first phase; it sends $n + m - 1$ rays as in fig 1, in second phase; it sends $n + m - 1$ rays as in fig 2. When a ray completes its cells, it calculates the summation of the integers in the cells it has visited. For example, you are given a 7×6 grid. Then the rays are:

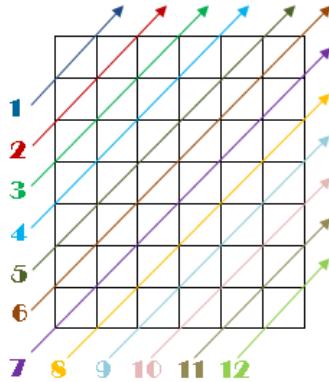


Fig 1: order of the rays in first phase

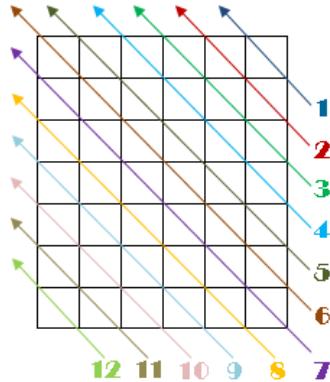


Fig 2: order of the rays in second phase

Each of the integers in the cells lies in the range [1, 100]. Now you are given the summation found by the rays in both phases, your task is to generate the grid.

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a line containing two integers: m and n ($1 \leq m, n \leq 50$). The next line contains $n + m - 1$ integers (space separated) denoting the summation found by the rays in phase 1. The next line contains $n + m - 1$ integers (space separated) denoting the summation found by the rays in phase 2. You can assume that these values are generated from a board which follows the restrictions described above.

Output

For each case, print the case number in a line. Then print the grid using the following format. Print m lines, each containing n integers, and two consecutive integers should be separated by a single space. Since there can be many solutions, print any valid one.

Sample Input	Output for Sample Input
1 2 3 2 8 11 5 9 12 4 1	Case 1: 2 7 9 1 2 5

Note

This is a special judge problem; wrong output format may cause 'wrong answer'.

1263 - Equalizing Money

There are **n** people in a certain village. Each of them contains some amount of money. One day a wise person told them to distribute the money such that everyone has equal amount of money. If they can do so, they will be favored by their fortunes.

You are given the information about the money of each person and some relations. Each relation is of the form **u v**. That means person **u** and **v** are capable of making money transactions. They are allowed to use transactions any number of times but they have to do integer transactions only.

Now your task is to answer whether they can redistribute the money such that each of them contains exactly same of money.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($2 \leq n \leq 1000$) and **m** ($0 \leq m \leq 10000$) where **m** denotes the number of relations. The next line contains **n** space separated integers ranging from **0** to **1000**. The **ith** integer of this line denotes the money for the **ith** person. Each of the next **m** lines contains two integers **u v** ($1 \leq u, v \leq n, u \neq v$) meaning that person **u** and **v** can make transactions. No relation is reported more than once.

Output

For each case, print the case number and 'Yes' if they can equalize their money or 'No' otherwise.

Sample Input	Output for Sample Input
3 5 4 1 0 1 1 2 1 2 2 3 3 4 4 5 2 1 5 10 1 2 4 2 1 1 0 2 1 2 2 3	Case 1: Yes Case 2: No Case 3: No

Note

Dataset is huge, use faster I/O methods.

1264 - Grouping Friends

There were n friends. But some of them had problems with others. So, we can say each person has some dissatisfaction with others. And hence we introduce a new term 'Dissatisfaction Factor' and it is described as follows:

You are given all the dissatisfaction factors as d_{ij} . d_{ij} means the dissatisfaction factor according to person i towards j . If the value is negative that means i^{th} person likes j^{th} person. Positive value means i^{th} person hates j^{th} person. 0 means i^{th} person is neutral about j^{th} person. Obviously d_{ij} can be different from d_{ji} since i^{th} person may like j^{th} person, but j^{th} person may not like i^{th} person and vice versa.

Now, if you group some people, the dissatisfaction factor is the summation of all the members' dissatisfaction factors towards other people in the group. You have to group them with minimum dissatisfaction factor. You can make as many groups as you like.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer n ($2 \leq n \leq 14$). Each of the next n lines contains n space separated integers. The j^{th} integer in the i^{th} line denotes d_{ij} . You can assume that $-100 \leq d_{ij} \leq 100$ and of course $d_{ii} = 0$.

Output

For each case, print the case number and the minimum dissatisfaction factor you can make after grouping them.

Sample Input	Output for Sample Input
3 3 0 2 3 -3 0 5 2 3 0 4 0 -1 -2 -3 1 0 -2 3 1 2 0 -1 2 -2 1 0 2 0 5 1 0	Case 1: -1 Case 2: -2 Case 3: 0

1265 – Island of Survival

You are in a reality show, and the show is way too real that they threw you into an island. Only two kinds of animals are in the island, the tigers and the deer. Though unfortunate but the truth is that, each day exactly two animals meet each other. So, the outcomes are one of the following

- a) If you and a tiger meet, the tiger will surely kill you.
- b) If a tiger and a deer meet, the tiger will eat the deer.
- c) If two deer meet, nothing happens.
- d) If you meet a deer, you may or may not kill the deer (depends on you).
- e) If two tigers meet, they will fight each other till death. So, both will be killed.

If in some day you are sure that you will not be killed, you leave the island immediately and thus win the reality show. And you can assume that two animals in each day are chosen uniformly at random from the set of living creatures in the island (including you).

Now you want to find the expected probability of you winning the game. Since in outcome (d), you can make your own decision, you want to maximize the probability.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a line containing two integers **t ($0 \leq t \leq 1000$)** and **d ($0 \leq d \leq 1000$)** where **t** denotes the number of tigers and **d** denotes the number of deer.

Output

For each case, print the case number and the expected probability. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 0 0 1 7 2 0 0 10	Case 1: 1 Case 2: 0 Case 3: 0.3333333333 Case 4: 1

1266 – Points in Rectangle

As the name says, this problem is about finding the number of points in a rectangle whose sides are parallel to axis. All the points and rectangles consist of 2D Cartesian co-ordinates. A point that lies in the boundary of a rectangle is considered inside.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a line containing an integer **q ($1 \leq q \leq 30000$)** denoting the number of queries. Each query is either one of the following:

- 1) **0 x y**, meaning that you have got a new point whose co-ordinate is **(x, y)**. But the restriction is that, if a point **(x, y)** is already listed, then this query has no effect.
- 2) **1 x₁ y₁ x₂ y₂** meaning that you are given a rectangle whose lower left co-ordinate is **(x₁, y₁)** and upper-right corner is **(x₂, y₂)**; your task is to find the number of points, given so far, that lie inside this rectangle. You can assume that **(x₁ < x₂, y₁ < y₂)**.

You can assume that the values of the co-ordinates lie between **0** and **1000** (inclusive).

Output

For each case, print the case number in a line first. Then for each query type (2), you have to answer the number of points that lie inside that rectangle. Print each of the results in separated lines.

Sample Input	Output for Sample Input
1 9 0 1 1 0 2 6 1 1 1 6 6 1 2 2 5 5 0 5 5 1 0 0 6 5 0 3 3 0 2 6 1 2 1 10 10	Case 1: 2 0 2 3

Note

Dataset is huge, use faster I/O methods.

1267 - Points in Rectangle (II)

As the name says, this problem is about finding the number of points in a rectangle whose sides are parallel to axis. All the points and rectangles consist of 2D Cartesian co-ordinates. A point that lies in the boundary of a rectangle is considered inside.

Initially you are given a list of points, after that you are given some queries, each query contains a rectangle. Your task is to find the number of points that lie in each of the query rectangle.

Input

Input starts with an integer **T (≤ 5)**, denoting the number of test cases.

Each case starts with a line containing two integers: **p, q ($1 \leq p, q \leq 50000$)**, where **p** denotes the number of points and **q** denotes the number of query rectangles.

Each of the next **p** lines contains two integers **x y** denoting the co-ordinate of a point. All the points are distinct.

Each of the next **q** lines contains four integers **x₁ y₁ x₂ y₂** meaning that you are given a rectangle whose lower left co-ordinate is **(x₁, y₁)** and upper-right corner is **(x₂, y₂)**. You can assume that **(x₁ < x₂, y₁ < y₂)**.

You can assume that the values of the co-ordinates lie between **0** and **10⁹** (inclusive).

Output

For each case, print the case number in a line first. Then for each query rectangle, you have to answer the number of points that lie inside that rectangle. Print each of the results in separate lines.

Sample Input	Output for Sample Input
1 5 6 0 0 5 9 2 3 4 6 7 7 0 0 5 9 2 2 10 11 4 6 7 9 3 3 4 5 4 6 5 7 5 7 7 9	Case 1: 4 4 3 0 1 2

Note

Dataset is huge, use faster I/O methods.

1268 – Unlucky Strings

Mr. 'Jotishi' is a superstitious man. Before doing anything he usually draws some strange figures, and decides what to do next.

One day he declared that the names that contain a string **S** as substring is unlucky. For example, let **S** be 'ab', then '**abc**', '**cabe**', '**pqqab**', '**ab**' etc are unlucky but '**ba**', '**baa**' etc are not.

So, he gives you the string **S** and asks you to find the number of names of length **n**, which are lucky, that means you have to find the number of strings that don't contain **S** as substring.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 10^9$). The next line contains the allowed characters for a name. This non-empty line contains lowercase characters only and in ascending order. The next line contains a string **S** ($1 \leq \text{length}(S) \leq 50$), and **S** contains characters from the allowed characters only.

Output

For each case, print the case number and the total number of names that don't contain **S** as substring. As the number can be very large, print the number modulo 2^{32} .

Sample Input	Output for Sample Input
3 3 ab ab 4 acd ca 5 ab aaa	Case 1: 4 Case 2: 55 Case 3: 24

1269 – Consecutive Sum

Little Jimmy is learning how to add integers. As in decimal the digits are 0 to 9, it makes a bit hard for him to understand the summation of all pair of digits. Since addition of numbers requires the knowledge of adding digits. So, his mother gave him a software that can convert a decimal integer to its binary and a binary to its corresponding decimal. So, Jimmy's idea is to convert the numbers into binaries, and then he adds them and turns the result back to decimal using the software. It's easy to add in binary, since you only need to know how to add (0, 0), (0, 1), (1, 0), (1, 1). Jimmy doesn't have the idea of carry operation, so he thinks that

$$1 + 1 = 0$$

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$0 + 0 = 0$$

Using these operations, he adds the numbers in binary. So, according to his calculations,

$$3 (011) + 7 (111) = 4 (100)$$

Now you are given an array of **n** integers, indexed from **0** to **n-1**, you have to find two indices **i j** in the array (**0 ≤ i ≤ j < n**), such that the summation (according to Jimmy) of all integers between indices **i** and **j** in the array, is maximum. And you also have to find two indices, **p q** in the array (**0 ≤ p ≤ q < n**), such that the summation (according to Jimmy) of all integers between indices **p** and **q** in the array, is minimum. You only have to report the maximum and minimum integers.

Input

Input starts with an integer **T** (**≤ 10**), denoting the number of test cases.

Each case starts with a line containing an integer **n** (**1 ≤ n ≤ 50000**). The next line contains **n** space separated non-negative integers, denoting the integers of the given array. Each integer fits into a 32 bit signed integer.

Output

For each case, print the case number, the maximum and minimum summation that can be made using Jimmy's addition.

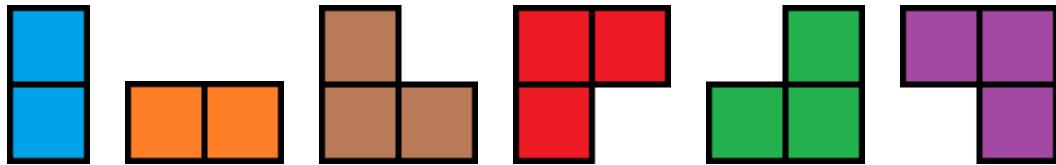
Sample Input	Output for Sample Input
2 5 6 8 2 4 2 5 3 8 2 6 5	Case 1: 14 2 Case 2: 15 1

Note

Dataset is huge, use faster I/O methods.

1270 – Tiles (II)

There is an $M \times N$ board, six types of tiles are available, and each of them is infinitely many, you have to find the number of ways you can fill the board using the tiles. Two board configurations are different if at least in one cell, their colors differ. The tiles are given below:



You **cannot** rotate or flip any tile. And no cell in the board should be empty. But some cells may be broken; you can't place any part of a tile in the broken cells. And there will be at least one cell which is not broken. The tiles shouldn't overlap.

For example, a 2×3 board can be colored by 5 ways, they are:



Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers: $M N$ ($1 \leq M, N \leq 100, \min(M, N) \leq 8$). Each of the next M lines contains N characters forming the board. There are two types of characters. A '.' means the cell is **not** broken; a '#' means the cell is **broken**.

Output

For each case, print the case number and the number of ways the board can be colored. The number may be large, so, output the number modulo 2^{64} .

Sample Input	Output for Sample Input
3 2 3 2 3 ...# ... 5 5	Case 1: 5 Case 2: 2 Case 3: 21272

1271 – Better Tour

Alice and Bob like to travel. One day Alice went for a tour, and after coming back she described his tour to Bob. Bob got quite excited hearing the adventures Alice made. So, he planned to make a tour like Alice. As Alice told him that in Alice's path the two most interesting places were the first city and the last city she visited. So, Bob tries to make a tour that contains minimum number of cities and which starts from the city where Alice started her journey, and which finishes in the city where Alice finished her journey.

But the problem is that Bob doesn't have the map, so, he doesn't know the information of the roads between cities. But he has Alice's diary, so he figured out the cities Alice visited in order. It's known that there are bidirectional roads connecting the cities. And each city is identified by an integer between **1** and **50000**.

Now Bob has given you the name of the cities visited by Alice in order, your task is to make a tour plan for Bob that visits **least** number of cities. Since there can be many such solutions, you have to find the tour which is lexicographically smallest. For example, let a tour be $c_1, c_2 \dots c_m$ and another tour be $d_1, d_2 \dots d_m$, (c_i, d_i denote cities), then the first tour is lexicographically smaller if

$c_1 < d_1$, or
 $c_1 = d_1$ and $c_2 < d_2$, or
 $c_1 = d_1$ and $c_2 = d_2$ and $c_3 < d_3$, or
...

Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($2 \leq n \leq 50000$). The next line contains **n** space separated integers denoting the tour made by Alice. Each of these integers will lie in the range **[1, 50000]**. Two adjacent integers: **u v** means that Alice moved to city **v** from city **u**. The starting city and the ending city will always be different. And two adjacent integers will also be different.

Output

For each case, print the case number in a single line. The next line should contain the tour plan for Bob. Two integers in this line should be separated by a single space.

Sample Input	Output for Sample Input
2 6 1 2 3 4 1 3 5 4 2 6 3 1	Case 1: 1 3 Case 2: 4 2 6 3 1

Note

Dataset is huge, use faster I/O methods.

1272 – Maximum Subset Sum

Little Jimmy is learning how to add integers. As in decimal the digits are 0 to 9, it makes a bit hard for him to understand the summation of all pair of digits. Since addition of numbers requires the knowledge of adding digits. So, his mother gave him a software that can convert a decimal integer to its binary and a binary to its corresponding decimal. So, Jimmy's idea is to convert the numbers into binaries, and then he adds them and turns the result back to decimal using the software. It's easy to add in binary, since you only need to know how to add (0, 0), (0, 1), (1, 0), (1, 1). Jimmy doesn't have the idea of carry operation, so he thinks that

$$\begin{aligned}1 + 1 &= 0 \\1 + 0 &= 1 \\0 + 1 &= 1 \\0 + 0 &= 0\end{aligned}$$

Using these operations, he adds the numbers in binary. So, according to his calculations,

$$3(011) + 7(111) = 4(100)$$

Now you are given an array of **n** integers, indexed from **0** to **n-1**, you have to find a subset of the integers in the array such that the summation (according to Jimmy) of all integers in the subset is as large as possible. You only have to report the maximum sum.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 100$). The next line contains **n** space separated non-negative integers, denoting the integers of the given array. Each integer fits into a 64 bit signed integer.

Output

For each case, print the case number, the maximum subset sum that can be made using Jimmy's addition.

Sample Input	Output for Sample Input
2 3 9 11 5 5 4 8 8 4 4	Case 1: 14 Case 2: 12

1273 - Strange Island

In a strange island there are two types of beings, 1) divine, 2) evil. Divines always speak the truth, Evils always lie. Unfortunately you are in this island and you are asked to find the minimum possible number of divines in the island. It's known that there is at least one divine in the island.

So, you are walking in the island, and when you meet a person you ask him a question that 'how many divines are in the island?' since you don't know whether he is a divine or evil. And the reply is always like, 'there are at least **p** divines in the island.'

Now, it's also known that you always meet a new person along your path. So, your task is to find the minimum possible number of divines in the island after having the answers from all the persons in the island.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 50000$)**, denoting the number of persons in the island. The next line contains **n** space separated integers between **1** and **n**. The **ith** integer in this line denotes the answer for the **ith** person you have met along your path.

Output

For each case, print the case number and the minimum possible number of divines in the island.

Sample Input	Output for Sample Input
2 3 1 1 3 4 1 2 3 4	Case 1: 2 Case 2: 1

Note

- 1) Dataset is huge, use faster I/O methods.
- 2) For sample 1, you met three persons, two of them said 'there are at least **1** divines in the island.' And the **3rd** person said, 'there are at least **3** divines in the island.' Since there is at least one divines in the island, so, the first and second persons are surely divines as they are speaking the truth. However, the **3rd** person may or may not be divine. So, the minimum possible number of divines in the island is **2**.

1274 - Beating the Dataset

You are in a contest, and unfortunately you don't have much time. You have one problem in hand; you just glanced at the sample output and found that it just wants 'YES' or 'NO'. So, you have made another plan instead of solving the problem as you know the system very well.

For this problem, every test case is stored in a separate file. When a submission is found, the system successively runs the solution on all tests of a problem, and for each test the checking process goes as follows. The input is copied to the file `input.txt`. Then the solution is launched. It reads the input from the file `input.txt` and writes the result to the file `output.txt`. When it finishes, the correct answer is copied to the file `answer.txt`. If the contents of the files `answer.txt` and `output.txt` match, the test is assumed to be passed; otherwise, the test is not passed.

So, you decided to write a program that would operate as follows. If the folder containing the program doesn't contain the file `answer.txt` (i.e. the program is run on the first test), then the program outputs "YES". Otherwise, the program outputs the contents of the file `answer.txt`. And before the contest, the sizes of the data files are given to you.

And it's clear that the size of the file with the answer "YES" is 3 bytes, the size of the file with the answer "NO" is 2 bytes, and all the variants of the order of tests are equally probable. Now you want to calculate the average number of tests that your solution won't pass.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a line containing two integers **n ($1 \leq n \leq 5000$)** and **s ($2n \leq s \leq 3n$)** where **n** denotes the number of data sets and **s** denotes the total size of the answer files.

Output

For each case, print the case number and the average number of tests your solution won't pass. Error less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 3 7 1 2 1 3 4 10	Case 1: 2 Case 2: 1 Case 3: 0 Case 4: 2.5000000000

Note

For the first case, one of the three answers is "YES" and two answers are "NO". If the order of tests is "YES-NO-NO", then your solution won't pass the second test only; if the order is "NO-YES-NO", then it will pass none of the tests; if the order is "NO-NO-YES", the solution won't pass the first and the third tests.

1275 – Internet Service Providers

A group of **N** Internet Service Provider companies (ISPs) use a private communication channel that has a maximum capacity of **C** traffic units per second. Each company transfers **T** traffic units per second through the channel and gets a profit that is directly proportional to the factor $T(C - T \cdot N)$. The problem is to compute the smallest value of **T** that maximizes the total profit the **N** ISPs can get from using the channel. Notice that **N**, **C**, **T**, and the optimal **T** are integer numbers.

Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

Each case starts with a line containing two integers **N** and **C** ($0 \leq N, C \leq 10^9$).

Output

For each case, print the case number and the minimum possible value of **T** that maximizes the total profit. The result should be an integer.

Sample Input	Output for Sample Input
6 1 0 0 1 4 3 2 8 3 27 25 1000000000	Case 1: 0 Case 2: 0 Case 3: 0 Case 4: 2 Case 5: 4 Case 6: 20000000

1276 – Very Lucky Numbers

According to some theory 4 and 7 are lucky digits, and all the other digits are not lucky. A lucky number is a number that contains only lucky digits in decimal notation. A very lucky number is a number that can be expressed as a product of several lucky numbers. A lucky number by itself is considered to be very lucky. For example, numbers 47, 49, 112 are very lucky.

Your task is to calculate the number of very lucky numbers that are not less than **A** and not greater than **B**.

Input

Input starts with an integer **T** (≤ 8000), denoting the number of test cases.

Each case starts with a line containing two integers **A** and **B** ($1 \leq A \leq B \leq 10^{12}$).

Output

For each case, print the case number and the result.

Sample Input	Output for Sample Input
4 1 2 88 99 112 112 1 100	Case 1: 0 Case 2: 0 Case 3: 1 Case 4: 10

Note

Very lucky numbers for the last sample input are 4, 7, 16, 28, 44, 47, 49, 64, 74 and 77.

1277 – Looking for a Subsequence

Given a sequence of integers $S = \{a_1, a_2, a_3 \dots a_n\}$ and an integer m , you have to find a subsequence of S containing m elements, $P = \{b_1, b_2, b_3 \dots b_m\}$ such that ($b_1 < b_2 < \dots < b_m$). If there are several subsequences possible then you should find the one where b_1 is leftmost. If there is still a tie, then check for the one where b_2 is leftmost and so on.

For example, let the sequence be, 8 7 5 6 5 1 2 7 and $m = 2$. Then (1, 2), (5, 6), (5, 7), (1, 7) ... etc can be solutions. But we are looking for (5, 6).

Input

Input starts with an integer T (≤ 12), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 10^5$) and q ($1 \leq q \leq 10$) where q denotes the total number of queries. The next line contains n space separated integers denoting $a_1, a_2 \dots a_n$ respectively. Each of the next q lines contains an integer m ($1 \leq m \leq n$). You can assume that $-10^8 \leq a_i \leq 10^8$.

Output

For each case, print the case number in a line. Then for each query print a single line containing the desired subsequence or 'Impossible' if there is no such subsequence. Insert a single space between two integers of a subsequence.

Sample Input	Output for Sample Input
3 6 3 3 4 1 2 3 6 6 4 2 6 2 2 4 6 1 3 5 3 4 8 5 8 7 5 6 5 1 2 7 1 2 3 4 5	Case 1: Impossible 1 2 3 6 3 4 Case 2: 2 4 6 Impossible Case 3: 8 5 6 5 6 7 Impossible Impossible

Note

Dataset is huge, use faster I/O methods.

1278 – Sum of Consecutive Integers

Given an integer **N**, you have to find the number of ways you can express **N** as sum of consecutive integers. You have to use at least two integers.

For example, **N = 15** has three solutions, (1+2+3+4+5), (4+5+6), (7+8).

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with a line containing an integer **N** ($1 \leq N \leq 10^{14}$).

Output

For each case, print the case number and the number of ways to express **N** as sum of consecutive integers.

Sample Input	Output for Sample Input
5	Case 1: 1
10	Case 2: 3
15	Case 3: 1
12	Case 4: 2
36	Case 5: 47
828495	

1279 – Graph Coloring

Given an undirected graph \mathbf{G} , you want to color each vertex of the graph using \mathbf{K} colors. The colors are numbered from 0 to $\mathbf{K}-1$. But there is one restriction. Let v be any vertex of the graph and $u_1, u_2 \dots u_m$ be the adjacent vertices of v , then

$$\text{color}(v) = \text{color}(u_1) + \text{color}(u_2) + \dots + \text{color}(u_m) \pmod K$$

Now you have to find the number of ways you can color the graph maintaining this restriction.

Input

Input starts with an integer T (≤ 200), denoting the number of test cases.

Each case starts with a line containing three integers N , M and K ($1 \leq N \leq 100$, $2 \leq K \leq 10^9$ and K is a prime) where N denotes the number of vertices and M denotes the number of edges of the graph. Each of the next M lines contains two integers u v ($1 \leq u, v \leq N$, $u \neq v$) denoting that there is an edge between vertex u and v . You can safely assume that there is at most one edge between any two vertices.

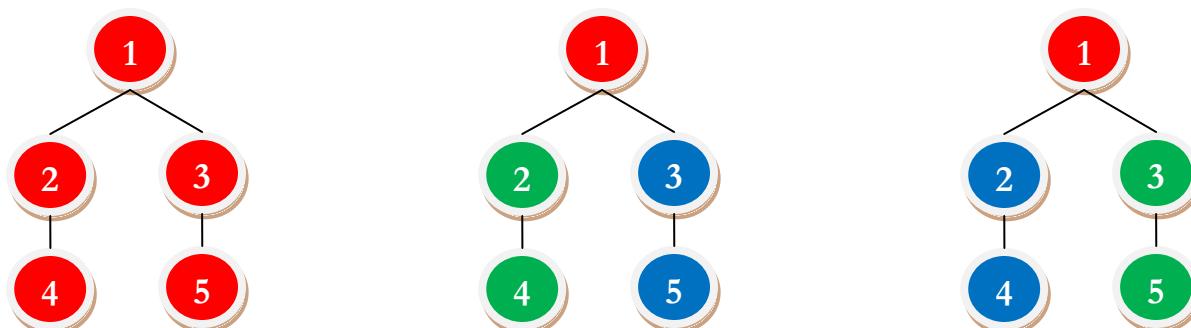
Output

For each case, print the case number and the result modulo 1000000007 .

Sample Input	Output for Sample Input
2 6 0 5 5 4 3 1 2 1 3 2 4 3 5	Case 1: 1 Case 2: 3

Note

For the second case, let the colors are red (0), green (1) and blue (2). Then the possible results are:



1280 – Best Grade

Your exam is over and you are about to get your grade. You have got marks in n subjects and for each subject you are given two integers p_i and m_i , p_i is the mark you obtained and m_i is the total mark in i^{th} subject. Now your teacher is about to remove d of the subjects, and after that he will calculate your average grade by the following rule. He first adds the total numbers that means the summation of all m_i of the remaining subjects, let this summation be M . And then he adds all the numbers you obtained in those subjects, let this summation be P . So, your grade is $(P/M) * 100$.

Now you asked your teacher to give you the permission to remove these d subjects by yourself. So, he gave you the opportunity to remove the subjects by your own. So, given all the marks, your task is to remove exactly d subjects such that your average grade becomes highest.

For example, there are three subjects and your marks are 5, 7 and 10 and the total marks in the subjects are 10, 20 and 15 respectively. And let d be 1. So, if you remove the second subject, it's better for you. Because then your grade becomes $(15/25*100)$ which is 60%.

Input

Input starts with an integer T (≤ 25), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers n ($2 \leq n \leq 20000$) and d ($1 \leq d < n$). Each of the next n lines contains two integers p_i and m_i ($1 \leq m_i \leq 1000$, $0 \leq p_i \leq m_i$) where p_i is the mark you obtained and m_i is the total mark in i^{th} subject.

Output

For each case, print the case number and the best average grade you can have. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 3 1 5 10 7 20 10 15 5 3 1 2 5 9 3 8 4 10 1 3	Case 1: 60 Case 2: 54.545454545

Note

Dataset is huge, use faster I/O methods.

1281 - New Traffic System

The country - Ajobdesh has a lot of problems in traffic system. As the Govt. is very clever (!), they made a plan to use only one way roads. Two cities **s** and **t** are the two most important cities in the country and mostly people travel from **s** to **t**. That's why the Govt. made a new plan to introduce some new one way roads in the traffic system such that the time to travel from **s** to **t** is reduced.

But since their budget is short, they can't construct more than **d** roads. So, they want to construct at most **d** new roads such that it becomes possible to reach **t** from **s** in shorter time. Unluckily you are one living in the country and you are assigned this task. That means you will be given the existing roads and the proposed new roads, you have to find the best path from **s** to **t**, which may allow at most **d** newly proposed roads.

Input

Input starts with an integer **T** (≤ 30), denoting the number of test cases.

Each case starts with a line containing four integers **n** ($2 \leq n \leq 10000$), **m** ($0 \leq m \leq 20000$), **k** ($0 \leq k \leq 10000$), **d** ($0 \leq d \leq 10$) where **n** denotes the number of cities, **m** denotes the number of existing roads and **k** denotes the number of proposed new roads. The cities are numbered from **0** to **n-1** and city **0** is denoted as **s** and city (**n-1**) is denoted as **t**.

Each of the next **m** lines contains a description of a road, which contains three integers **u_i v_i w_i** ($0 \leq u_i, v_i < n$, $u_i \neq v_i$, $1 \leq w_i \leq 1000$) meaning that there is a road from **u_i** to **v_i** and it takes **w_i** minutes to travel in the road. There is at most one road from one city to another city.

Each of the next **k** lines contains a proposed new road with three integers **u_i v_i w_i** ($0 \leq u_i, v_i < n$, $u_i \neq v_i$, $1 \leq w_i \leq 1000$) meaning that the road will be from **u_i** to **v_i** and it will take **w_i** minutes to travel in the road. There can be at most one proposed road from one city to another city.

Output

For each case, print the case number and the shortest path cost from **s** to **t** or "Impossible" if there is no path from **s** to **t**.

Sample Input	Output for Sample Input
2 4 2 2 2 0 1 10 1 3 20 0 2 5 2 3 14 2 0 1 0 0 1 100	Case 1: 19 Case 2: Impossible

Note

Dataset is huge, use faster I/O methods.

1282 - Leading and Trailing

You are given two integers: n and k , your task is to find the most significant three digits, and least significant three digits of n^k .

Input

Input starts with an integer T (≤ 1000), denoting the number of test cases.

Each case starts with a line containing two integers: n ($2 \leq n < 2^{31}$) and k ($1 \leq k \leq 10^7$).

Output

For each case, print the case number and the three leading digits (most significant) and three trailing digits (least significant). You can assume that the input is given such that n^k contains at least six digits.

Sample Input	Output for Sample Input
5 123456 1 123456 2 2 31 2 32 29 8751919	Case 1: 123 456 Case 2: 152 936 Case 3: 214 648 Case 4: 429 296 Case 5: 665 669

1283 – Shelving Books

You are a librarian. You keep the books in a well organized form such that it becomes simpler for you to find a book and even they look better in the shelves.

One day you get **n** new books from one of the library sponsors. And unfortunately they are coming to visit the library, and of course they want to see their books in the shelves. So, you don't have enough time to shelve them all in the shelf in an organized manner since the heights of the books may not be same. But it's the question of your reputation, that's why you have planned to shelve them using the following idea:

- 1) You will take one book from the **n** books from left.
- 2) You have exactly one shelf to organize these books, so you may either put this book in the left side of the shelf, right side of the shelf or you may not put it in the shelf. There can already be books in the left or right side. In such case, you put the book with that book, but you don't move the book you put previously.
- 3) Your target is to put the books in the shelf such that from left to right they are sorted in **non-descending** order.
- 4) Your target is to put as many books in the shelf as you can.

You can assume that the shelf is wide enough to contain all the books. For example, you have 5 books and the heights of the books are 3 9 1 5 8 (from left). In the shelf you can put at most 4 books. You can shelve 3 5 8 9, because at first you got the book with height 3, you stored it in the left side of the shelf, then you got 9 and you put it in the right side of the shelf, then you got 1 and you ignored it, you got 5 you put it in the left with 3. Then you got 8 and you put it in left or right. You can also shelve 1 5 8 9 maintaining the restrictions.

Now given the heights of the books, your task is to find the maximum number of books you can shelve maintaining the above restrictions.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 100$)**. Next line contains **n** space separated integers from **[1, 10^5]**. The **ith** integer denotes the height of the **ith** book from left.

Output

For each case, print the case number and the maximum number of books that can be shelved.

Sample Input	Output for Sample Input
2 5 3 9 1 5 8 8 121 710 312 611 599 400 689 611	Case 1: 4 Case 2: 6

1284 - Lights inside 3D Grid

You are given a 3D grid, which has dimensions **X**, **Y** and **Z**. Each of the **X x Y x Z** cells contains a light. Initially all lights are off. You will have **K** turns. In each of the **K** turns,

1. You select a cell **A** randomly from the grid,
2. You select a cell **B** randomly from the grid and
3. Toggle the states of all the bulbs bounded by cell **A** and cell **B**, i.e. make all the ON lights OFF and make all the OFF lights ON which are bounded by **A** and **B**. To be clear, consider cell **A** is (x_1, y_1, z_1) and cell **B** is (x_2, y_2, z_2) . Then you have to toggle all the bulbs in grid cell (x, y, z) where $\min(x_1, x_2) \leq x \leq \max(x_1, x_2)$, $\min(y_1, y_2) \leq y \leq \max(y_1, y_2)$ and $\min(z_1, z_2) \leq z \leq \max(z_1, z_2)$.

Your task is to find the expected number of lights to be ON after **K** turns.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

Each case starts with a line containing four integers **X**, **Y**, **Z** ($1 \leq X, Y, Z \leq 100$) and **K** ($0 \leq K \leq 10000$).

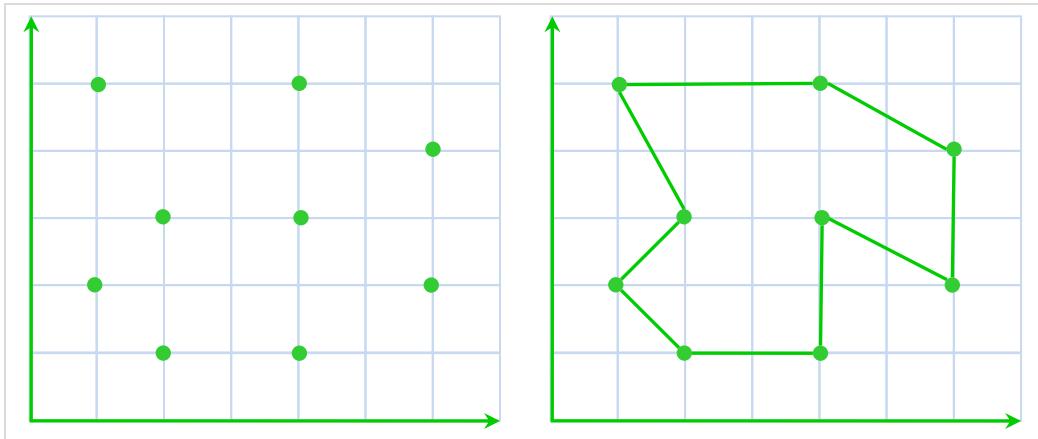
Output

For each case, print the case number and the expected number of lights that are ON after **K** turns. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
5 1 2 3 5 1 1 1 1 1 2 3 0 2 3 4 1 2 3 4 2	Case 1: 2.9998713992 Case 2: 1 Case 3: 0 Case 4: 6.375 Case 5: 9.09765625

1285 - Drawing Simple Polygon

Given set of points in the plane, your task is to draw a polygon using the points. You have to use all the points. To be more specific, each point of the set has to be a vertex of the polygon, and the polygon must not have any other vertices. No two line segments of the polygon may have any point in common, except for the middle vertex of two consecutive line segments. For example, given the points on the left-hand side, a valid polygon is shown on the right-hand side:



You can assume that, no two points will share the same location.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($3 \leq n \leq 2000$), denoting the number of points. The next line contains the co-ordinates of the points. Each point is specified by two integer **x y** in the range $[-10^4, 10^4]$.

Output

For each case, print the case number in a single line first. In the next line print '**Impossible**' if no solution can be found. Otherwise print a permutation of the numbers **0** to **n-1**. Each of these numbers represents the index of a point, in the same order as given in the input. When drawing line segments between consecutive points in the order given by this permutation, the result must be a valid polygon. Insert a single space between two integers.

Sample Input	Output for Sample Input
2 4 0 0 2 0 0 1 1 0 5 0 0 10 0 10 5 5 -1 0 5	Case 1: 0 3 1 2 Case 2: 2 1 3 0 4

Note

This is a special judge problem; wrong output format may cause '**wrong answer**'.

1286 – Space Shuttle Experiments

Professor Spook is consulting for NASA, which is planning a series of space shuttle flights and must decide which commercial experiments to perform and which instruments to have on board each flight. For each flight NASA considers a set $E = \{E_1, E_2, \dots, E_m\}$ of instruments experiments and the commercial sponsor of E_j has agreed to pay NASA p_j dollars for the results of the experiments.

The experiments use a set $I = \{I_1, I_2, \dots, I_n\}$ of instruments; each experiment E_j requires some of the instruments from the set. The cost of carrying instruments I_k is c_k dollars. And an instrument can be used for multiple experiments.

The professor's job is to determine which experiments to perform and which instruments to carry for a given flight in order to maximize the net revenue, which is the total income from the experiments performed minus the total cost of the instruments carried. Since he is not a programmer, he asked your help.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers m ($1 \leq m \leq 100$) and n ($1 \leq n \leq 100$), where m denotes the number of experiments and n denotes the number of instruments. The next line contains m space separated integers, where the j^{th} integer denotes the commercial sponsor of E_j , paying NASA p_j ($1 \leq p_j \leq 10000$) dollars for the result of the experiment. The next line contains n space separated integers, where the k^{th} integer denotes the cost of carrying the k^{th} instrument, c_k ($1 \leq c_k \leq 10000$). Each of the next m lines contains an integer q_i ($1 \leq q_i \leq n$) followed by q_i distinct integers each between 1 and n , separated by spaces. These q_i integers denote the required instruments for the i^{th} experiment.

Output

For each case, print the case number and the maximum revenue NASA can make using the experiments.

Sample Input	Output for Sample Input
2 1 1 10 20 1 1 3 5 20 30 40 1 2 30 4 50 3 1 2 3 3 2 3 4 1 5	Case 1: 0 Case 2: 13

1287 – Where to Run

Last night you robbed a bank but couldn't escape and when you just got outside today, the police started chasing you. The city, where you live in, consists of some junctions which are connected by some bidirectional roads.

Since police is behind, you have nothing to do but to run. You don't know whether you would get caught or not, but if it is so, you want to run as long as you can. But the major problem is that if you leave a junction, next time you can't come to this junction, because a group of police wait there for you as soon as you left it, while some other keep chasing you.

That's why you have made a plan to fool the police as longer time as possible. The plan is, from your current junction, you first find the number of junctions which are safe (no police are there) and if you go to one of them; you are still able to visit all the safe junctions (in any order) maintaining the above restrictions. You named them 'Elected Junction' or **EJ**. If there is no such junction; you stop running, because you lose your mind thinking what to do, and the police catch you immediately.



But if there is at least one EJ, you can either fool around the police by staying in the current junction for 5 minutes (actually you just hide there, so the police lose your track thinking which road you might have taken), or you can choose to go to any EJ. The probability of choosing to stay in the current junction or to go to each of the EJ is equal. For example, from the current junction you can go to three EJs, that means the probability of staying in the current junction is **1/4** or the probability to go to any of the EJ is **1/4** since you have four options (either stay in the current junction or go to any of the three junctions).

You can fool the police (by hiding) multiple times in a city, but of course the above conditions should be satisfied. And you have decided not to stop in the middle of any road, because you have the fear that, if you stop in the middle of any road, then the police would surround you from both ends.

Now, given the map of the city and the required time for you to travel in each road of the map; you have to find the expected time for the police to catch you.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n ($1 \leq n \leq 15$)** denoting the number of junctions and **m**, denoting the number of roads in the city. The junctions are numbered from **0** to **n - 1**.

Each of the next **m** lines contains three integers **u v w** ($0 \leq u, v < n, 0 < w \leq 100, u \neq v$) meaning that there is a road between junction **u** and **v** and you need **w** minutes to travel in the road. Your home is in junction **0** and you are initially in your home. And you may safely assume that there can be at most one road between a pair of junctions.

Output

For each case, print the case number and the expected time in minutes. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
<pre> 3 3 2 0 1 3 1 2 3 4 6 0 1 75 0 2 86 0 3 4 1 2 1 1 3 53 2 3 10 5 5 0 1 10 1 2 20 2 3 30 1 3 20 3 4 10 </pre>	<pre> Case 1: 16 Case 2: 106.8333333333 Case 3: 90 </pre>

Note

For the 3rd case, initially you are in junction 0, and you can either stay here for 5 minutes, or you can move to 1. The probability of staying in 0 is **0.5** and the probability of going to junction 1 is also **0.5**. Now if you are in junction 1, either you can stay here for 5 minutes or you can move to junction 2. From junction 1, you cannot move to junction 3, because if you go to junction 3, you can move to junction 2 or junction 4, but if you go to 2, you cannot visit junction 4 (since police would have occupied junction 3), and if you go to junction 4 from 3, you cannot visit junction 2 for the same reason. So, from 1, junction 2 is the only **EJ**, but junction 3 is not.

1288 - Subsets Forming Perfect Squares

You are given a set of **n** integers, and you want to take a subset of the integers, containing at least one integer, such that the multiplication of them forms a perfect square. An integer **y** is said to be a **perfect square** if **y** can be written as $(x * x)$ where **x** is an integer. For example, 36 ($6*6$), 49 ($7*7$) etc are perfect squares.

For example, the given integers are 2, 3, 5, 10 and 6, then the resulting subsets are {2, 3, 6}, {2, 5, 10} and {2, 3, 5, 6, 10}. For 4, 5 and 5 the resulting subsets are {4}, {5, 5} and {4, 5, 5}. For 4 and 4 the resulting subsets are {4}, {4} and {4, 4}.

Now your task is to find the number of ways to take the subsets such that the multiplication of a subset makes a perfect square.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 1000$). The next line contains **n** space separated integers denoting the set. You can assume that each of the integers lies in the range $[2, 10^{15}]$ and no integer contains a prime factor greater than 300.

Output

For each case, print the case number and the total number of such subsets. Since the result can be very large, print the result modulo **1000000007**.

Sample Input	Output for Sample Input
6 5 2 3 5 10 6 3 4 5 5 2 4 4 4 5 5 25 20 3 2 3 4 5 2 6 5 7 11	Case 1: 3 Case 2: 3 Case 3: 3 Case 4: 7 Case 5: 1 Case 6: 0

1289 – LCM from 1 to n

Given an integer **n**, you have to find

lcm(1, 2, 3, ..., n)

lcm means least common multiple. For example $\text{lcm}(2, 5, 4) = 20$, $\text{lcm}(3, 9) = 9$, $\text{lcm}(6, 8, 12) = 24$.

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($2 \leq n \leq 10^8$).

Output

For each case, print the case number and **lcm(1, 2, 3, ..., n)**. As the result can be very big, print the result modulo 2^{32} .

Sample Input	Output for Sample Input
5	Case 1: 2520
10	Case 2: 60
5	Case 3: 2300527488
200	Case 4: 360360
15	Case 5: 232792560
20	

1290 - The Great Merger

Woohoo and MarcoHard are two prominent software companies. But due to the rise of rival firms, they decided that they need to merge in order to compete with them. For this, the management in Woohoo and MarcoHard are facing some troubles.

Both Woohoo and MarcoHard maintain a hierarchy of employees. In both companies, the head of the company is the CEO. Every other employee has exactly one immediate superior.

To make the merger successful, they have to maintain exactly same hierarchy in both the companies. They decided not to fire any of the employees. They can change the name of the employee's post (e.g. one company has a post for 'Software Developer', in other company it is called 'Code Ninja'), but they can't promote or demote anyone (e.g. they can't make Developer a Project Manager or vice versa), nor can they break the existing hierarchy (that is if **a** is an immediate superior of **b**, they should be the same in the new hierarchy). But they can make new posts and hire new employees. Now, they need to make a similar hierarchy by hiring minimum number of employees. Since, they are people in HR, and not much familiar with computing algorithms, they are finding it really tough to figure out how many people to hire. So, you are asked to this job for them. And guess what, you may be awarded a balloon if you can help them in this great dilemma.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each test case contains a blank line and two more lines, the hierarchy of WooHoo and Marcohard. Each hierarchy description starts with an integer **N** ($1 \leq N \leq 100$), the number of employees except the CEO, followed by **N** integers, each describing the immediate superior of that employee. The CEO is identified by **0th** employee and every other employee is identified by an integer between **1** and **N**.

You can assume that, the hierarchy is valid, that is every employee is direct or indirect subordinate of the CEO and no one is a direct or indirect superior of himself.

Output

For each test case, print the case number and the minimum number employees to hire. Both WooHoo and MarcoHard can hire, so, this number will be the total number of new employees hired by these two companies.

Sample Input	Output for Sample Input
2 4 0 0 1 1 4 0 1 2 2 4 0 0 1 1 4 0 0 2 2	Case 1: 4 Case 2: 0

1291 – Real Life Traffic

Dhaka city is full of traffic jam and when it rains, some of the roads become unusable. So, you are asked to redesign the traffic system of the city such that if exactly one of the roads becomes unusable, it's still possible to move from any place to another using other roads.

You can assume that Dhaka is a city containing some places and bi directional roads connecting the places and it's possible to go from any place to another using the roads. There can be at most one road between two places. And of course there is no road that connects a place to itself. To be more specific there are **n** places in Dhaka city and for simplicity, assume that they are numbered from **0** to **n-1** and there are **m** roads connecting the places.

Your plan is to build some new roads, but you don't want to build a road between two places where a road already exists. You want to build the roads such that if any road becomes unusable, there should be an alternate way to go from any place to another using other roads except that damaged road. As you are a programmer, you want to find the minimum number of roads that you have to build to make the traffic system as stated above.

Input

Input starts with an integer **T** (≤ 30), denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers: **n** ($3 \leq n \leq 10000$) and **m** (≤ 20000). Each of the next **m** lines contains two integers **u v** ($0 \leq u, v < n, u \neq v$) meaning that there is a bidirectional road between place **u** and **v**. The input follows the above constraints.

Output

For each case, print the case number and the minimum number of roads you have to build such that if one road goes down, it's still possible to go from any place to another.

Sample Input	Output for Sample Input
2 4 3 1 2 2 3 2 0 3 3 1 2 2 0 0 1	Case 1: 2 Case 2: 0

Note

1. Dataset is huge, use faster I/O methods.
2. For case 1, one of the solutions is to construct two roads in (0, 1) and (1, 3).

1292 – Laser Shot

You have a powerful laser gun and obstacles can be destroyed with the gun, and the laser never stops. Yes, it means that it destroys obstacles and keeps going. No obstacles can stop this.

Now you have exactly one laser shot left and you want to destroy maximum number of obstacles. The obstacles are scattered, and for this problem assume that the obstacles are placed in 2D Cartesian coordinates. You can move to any point as you want (not necessarily in integer coordinates), and you can make exactly one shot. Your target is to find the maximum number of obstacles you can destroy in this shot.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 700$)** denoting the number of points. Each of the next **n** lines contains two integers **x y ($-10000 \leq x, y \leq 10000$)** denoting the coordinate of an obstacle. You can assume that the coordinates of all the obstacles are distinct.

Output

For each case, print the case number and maximum number of obstacles you can destroy in one laser shot.

Sample Input	Output for Sample Input
2 5 1 1 2 2 2 9 3 10 10 17 3 5 6 6 7 2 8	Case 1: 3 Case 2: 2

1293 - Document Analyzer

You work in a leading software development company. As you are great in coding, most of the critical tasks are allotted for you. You like the challenge and you feel excited to solve those problems.

Recently your company is developing a project named Document Analyzer. In this project you are assigned a task; of course a critical task. The task is that you are given a document consisting of lowercase letters, numbers and punctuations. You have to analyze the document and separate the words first. Words are consecutive sequences of lower case letters. After listing the words, in the order same as they occurred in the document, you have to number them from **1, 2, ..., n**. After that you have to find the range **p** and **q** ($p \leq q$) such that all kinds of words occur between **p** and **q** (inclusive). If there are multiple such solutions you have to find the one where the difference of **p** and **q** is smallest. If still there is a tie, then find the solution where **p** is smallest.

Input

Input starts with an integer **T** (≤ 15), denoting the number of test cases.

Each case will be denoted by one or more lines denoting a document. Each line contains no more than **100** characters. A document will contain either lowercase letters or numbers or punctuations. The last line of a document will contain the word '**END**' which is of course not the part of the document. You can assume that a document will contain between **1** and **50000** words (inclusive). Words may contain up to **10** characters. And a document can contain up to **5000** lines.

Output

For each case, print the case number and **p** and **q** as described above.

Sample Input	Output for Sample Input
3 1. a case is a case, 2. case is not a case~ END a b c d e END a@#\$a^%a a a b b---b b++12b END	Case 1: 6 9 Case 2: 1 5 Case 3: 5 6

Note

Dataset is huge, use faster I/O methods.

1294 – Positive Negative Sign

Given two integers: **n** and **m** and **n** is divisible by **2m**, you have to write down the first **n** natural numbers in the following form. At first take first **m** integers and make their sign negative, then take next **m** integers and make their sign positive, the next **m** integers should have negative signs and continue this procedure until all the **n** integers have been assigned a sign. For example, let **n** be **12** and **m** be **3**. Then we have

-1 -2 -3 +4 +5 +6 -7 -8 -9 +10 +11 +12

If **n = 4** and **m = 1**, then we have

-1 +2 -3 +4

Now your task is to find the summation of the numbers considering their signs.

Input

Input starts with an integer **T (≤ 10000)**, denoting the number of test cases.

Each case starts with a line containing two integers: **n** and **m ($2 \leq n \leq 10^9, 1 \leq m$)**. And you can assume that **n** is divisible by **2*m**.

Output

For each case, print the case number and the summation.

Sample Input	Output for Sample Input
2 12 3 4 1	Case 1: 18 Case 2: 2

1295 – Lighting System Design

You are given the task to design a lighting system for a huge conference hall. After doing a lot of calculation & sketching, you have figured out the requirements for an energy-efficient design that can properly illuminate the entire hall. According to your design, you need lamps of **n** different power ratings. For some strange current regulation method, all the lamps need to be fed with the same amount of current. So, each category of lamp has a corresponding voltage rating. Now, you know the number of lamps and cost of every single unit of lamp for each category. But the problem is that you are to buy equivalent voltage sources for all the lamp categories. You can buy a single voltage source for each category (each source is capable of supplying to infinite number of lamps of its voltage rating) and complete the design. But the accounts section of your company soon figures out that they might be able to reduce the total system cost by eliminating some of the voltage sources and replacing the lamps of that category with higher rating lamps. Certainly you can never replace a lamp by a lower rating lamp as some portion of the hall might not be illuminated then. You are more concerned about money-saving rather than energy-saving. Find the minimum possible cost to design the system.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 1000$). Each of the next **n** lines contains four integers **V_i**, **K_i**, **C_i** and **L_i** ($1 \leq V_i \leq 10^5$, $1 \leq K_i \leq 1000$, $1 \leq C_i \leq 10$, $1 \leq L_i \leq 100$). Here the integers in **ith** line have the following meaning.

1. **V_i** means the voltage rating,
2. **K_i** means the cost of a voltage source of this category,
3. **C_i** means the cost of a lamp of this category and
4. **L_i** means the number of required lamps of this category.

You can assume that the voltage rating for the categories will be distinct.

Output

For each case, print the case number and the minimum possible cost to design the system.

Sample Input	Output for Sample Input
1 3 100 500 10 20 120 600 8 16 220 400 7 18	Case 1: 778

1296 - Again Stone Game

Alice and Bob are playing a stone game. Initially there are **n** piles of stones and each pile contains some stone. Alice starts the game and they alternate moves. In each move, a player has to select any pile and should remove at least one and no more than half stones from that pile. So, for example if a pile contains 10 stones, then a player can take at least 1 and at most 5 stones from that pile. If a pile contains 7 stones; at most 3 stones from that pile can be removed.

Both Alice and Bob play perfectly. The player who cannot make a valid move loses. Now you are given the information of the piles and the number of stones in all the piles, you have to find the player who will win if both play optimally.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 1000$). The next line contains **n** space separated integers ranging in $[1, 10^9]$. The i^{th} integer in this line denotes the number of stones in the i^{th} pile.

Output

For each case, print the case number and the name of the player who will win the game.

Sample Input	Output for Sample Input
5 1 1 3 10 11 12 5 1 2 3 4 5 2 4 9 3 1 3 9	Case 1: Bob Case 2: Alice Case 3: Alice Case 4: Bob Case 5: Alice

1297 - Largest Box

In the following figure you can see a rectangular card. The width of the card is W and length of the card is L and thickness is zero. Four ($x \times x$) squares are cut from the four corners of the card shown by the black dotted lines. Then the card is folded along the magenta lines to make a box without a cover.



Given the width and height of the box, you will have to find the maximum volume of the box you can make for any value of x .

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case starts with a line containing two real numbers L and W ($0 < L, W < 100$).

Output

For each case, print the case number and the maximum volume of the box that can be made. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3	Case 1: 4.513804324
2 10	Case 2: 2.2268848896
3.590 2.719	Case 3: 33.412886
8.1991 7.189	

1298 – One Theorem, One Year

A number is **Almost-K-Prime** if it has exactly **K** prime numbers (not necessarily distinct) in its prime factorization. For example, $12 = 2 * 2 * 3$ is an **Almost-3-Prime** and $32 = 2 * 2 * 2 * 2 * 2$ is an **Almost-5-Prime** number. A number **X** is called **Almost-K-First-P-Prime** if it satisfies the following criterions:

1. **X** is an **Almost-K-Prime** and
2. **X** has **all and only** the first **P** ($P \leq K$) primes in its prime factorization.

For example, if **K=3** and **P=2**, the numbers $18 = 2 * 3 * 3$ and $12 = 2 * 2 * 3$ satisfy the above criterions. And $630 = 2 * 3 * 3 * 5 * 7$ is an example of **Almost-5-First-4-Prime**.

For a given **K** and **P**, your task is to calculate the summation of $\Phi(X)$ for all integers **X** such that **X** is an **Almost-K-First-P-Prime**.

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case starts with a line containing two integers **K** ($1 \leq K \leq 500$) and **P** ($1 \leq P \leq K$).

Output

For each case, print the case number and the result modulo **1000000007**.

Sample Input	Output for Sample Input
3 3 2 5 4 99 45	Case 1: 10 Case 2: 816 Case 3: 49939643

Note

1. In mathematics $\Phi(X)$ means the number of relatively prime numbers with respect to **X** which are smaller than **X**. Two numbers are relatively prime if their GCD (Greatest Common Divisor) is 1. For example, $\Phi(12) = 4$, because the numbers that are relatively prime to 12 are: 1, 5, 7, 11.
2. For the first case, **K = 3** and **P = 2** we have only two such numbers which are **Almost-3-First-2-Prime**, $18=2*3*3$ and $12=2*2*3$. The result is therefore, $\Phi(12) + \Phi(18) = 10$.

1299 – Fantasy Cricket

ICC World Cup 2011 has just finished. During the world cup, lots of cricket fans were playing an online game named "Fantasy Cricket".

The score board of fantasy cricket looks like the following image. After each match of the world cup the score board of fantasy cricket updates.

Rank	Manager	Team Name	Total	
1	▲	Mind the Gap	Mind the Gap XI	12384
2	▼	Old Man	Old Man XI	12344
3	▲	Legend	Legend XI	11611
4	▼	Far Cry	Far Cry XI	11221
5	►	Accepted	Accepted XI	10111
6	▲	WA	WA XI	10001

Figure 1

Each player plays a role of a manager here. In the rank list there is a symbol besides each manager. There are three kinds of symbols. These are explained in the following table.

Symbol	Explanation	ASCII Representation
▲	The rank of the player has upgraded after last match, i.e. if the current rank of the player is K , the rank of the player before the last match was greater than K .	U
▼	The rank of the player has downgraded after the last match, i.e. if the current rank of the player is K , the rank of the player before the last match was less than K .	D
►	The rank of the player has not changed after the last match, i.e. if the current rank of the player is K , the rank of the player before the last match was also K .	E

You will be given such a score board after a particular match. Can you determine any possible valid ordering of the players exactly before that round? For this problem you have to print the number of possible ordering before the last match.

Here is an example –

Rank	Manager
1	A
2	B
3	C
4	D

Figure 2

Possible Previous Order 1	Possible Previous Order 2
B	B
A	D
D	A
C	C

Figure 3

For this rank (figure 2), only two different ordering are possible (figure 3) before the last match which comply the current ordering with the symbols.

Name of the managers are not important for this problem. So, for a scoreboard, you will be given a sequence of ASCII representation of the symbols (stated above), i.e. you will be given a string which only contains '**U**', '**D**' and '**E**'.

Input

Input starts with an integer **T** (≤ 300), denoting the number of test cases.

Each case starts with a line containing a string. The length of the string will be between **1** and **1000**. The string will contain characters from {**'U'**, **'D'**, **'E'**}.

Output

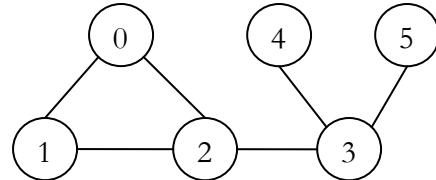
For each case, print the case number and the number of possible orderings modulo **1000000007**.

Sample Input	Output for Sample Input
3 UDUD EEE DU	Case 1: 2 Case 2: 1 Case 3: 0

1300 - Odd Personality

Being an odd person, Jim always liked odd numbers. One day he was visiting his home town which consists of **n** places and **m** bidirectional roads. A road always connects two different places and there is at most one road between two places. The places are numbered from **0** to **n-1**.

Jim wants to find a tour which starts from a place **p** and each time it goes to a new **road** and finally at last step it returns back to **p**. As Jim likes odd numbers, he wants the length of the tour to be odd. And the length of a tour is defined by the number of roads used in the tour.



For the city map given above, **0 - 1 - 2 - 0** is such a tour, so, **0** is one of the results, since from **0**, a tour of odd length is found, similarly, **1 - 2 - 0 - 1** is also a valid tour. But **3 - 2 - 0 - 1 - 2 - 3** is not. Since the road **2 - 3** is used twice. Now given the city map, Jim wants to find the number of places where he can start his journey for such a tour. As you are the best programmer in town, he asks you for help. Jim can use a place more than once, but a road can be visited at most once in the tour.

Input

Input starts with an integer **T** (≤ 30), denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers: **n** ($3 \leq n \leq 10000$) and **m** ($0 \leq m \leq 20000$). Each of the next **m** lines contains two integers **u v** ($0 \leq u, v < n, u \neq v$) meaning that there is a bidirectional road between place **u** and **v**. The input follows the above constraints. And no road is reported more than once.

Output

For each case, print the case number and the total number places where Jim can start his journey.

Sample Input	Output for Sample Input
1 6 6 0 1 1 2 2 0 3 2 3 4 3 5	Case 1: 3

Note

Dataset is huge, user faster I/O methods.

1301 – Monitoring Processes

We have planned to design a new operating system. Like other OS we will use the techniques of processes, schedulers, locks etc. The basic plan is to use the OS in hardwires that have low configurations. So, efficiency matters. That's why we want to minimize the cost as well as the power consumption. To be more specific, there are n processes, and each process starts its execution in timestamp s_i , and ends its execution in timestamp t_i . For simplicity assume that the timestamps are represented as integers. Now when a process is being executed, we need a wrapper program to look after the process. The reason behind using wrapper programs is that, they will continuously check the processes and if any process tries to harm the system or wants to take hold of some restricted resources or even tries to invoke some forbidden methods, the wrapper will halt the process and generate appropriate error signals. But the problem is that a wrapper program cannot monitor more than one process in any timestamp and when it's been assigned to a process, it will have to wait until the process finishes. But after this, the same wrapper program can be used for monitoring another process. So, a wrapper program can be used for multiple processes but not more than one in any timestamp.

So, we have the process schedules and we want to find the number of wrapper programs to monitor them according to the given restrictions. As you are the leading programmer of this project, you are asked to find the minimum number of wrapper programs to monitor all the processes.

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 50000$). Each of the next n lines contains two integers s_i t_i ($1 \leq s_i \leq t_i \leq 10^9$).

Output

For each case, print the case number and the minimum number of wrapper programs to monitor all the processes.

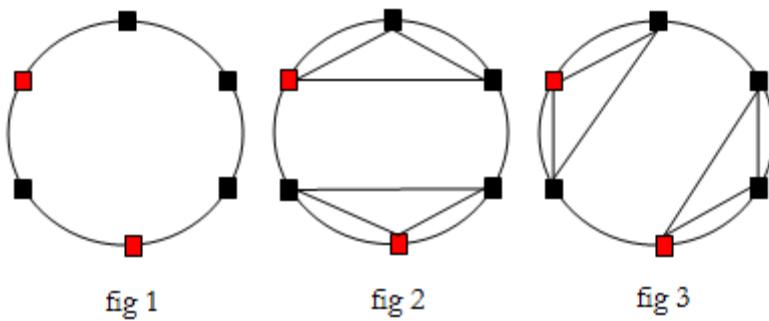
Sample Input	Output for Sample Input
2 2 1 3 3 5 4 1 10 10 20 11 21 3 5	Case 1: 2 Case 2: 2

Note

Dataset is huge, use faster I/O methods.

1302 – Independent Attacking Zones

A common technique used by invading armies is to surround a city instead of directly entering it. The armies divided themselves into platoons having bases in a circular fashion around the city. To take internal control of the city, platoons are grouped in three to cover triangular regions. It is a policy of the General to ensure that no two triangular regions overlap. Unfortunately, the process is made a bit trickier because there are two types of armies in the invading force. The two different armies are known as Red Army and Black Army. A platoon consists of one type of army. While the Black Army has clear intention to serve the General but the Red ones might betray if they get an opportunity. It is decided that every triangular group will consist of at most one Red Army Platoon so that the Red ones cannot dominate in any assignment. Suppose we have 6 platoons (4 black and 2 red) as shown in fig 1. We have only two valid arrangements, shown in fig 2 and fig 3.



You will be given the number of platoons, their positions and colors. You have to find the number of possible configurations such that every platoon is part of exactly one group and also meets the above restrictions.

Input

Input starts with an integer **T** (≤ 125), denoting the number of test cases.

Each case starts with a line containing a non empty string. Each character of the string is either an '**R**' or '**B**'. The string gives the position of the platoons in clockwise order. '**R**' indicates red and '**B**' indicates black. The starting position is arbitrarily chosen. So, the example above may be represented by any of the following: '**RBBRB**', '**BBBRBR**', '**BBRBRB**', '**BRBRBB**', '**RBRBBB**' or '**BRBBBB**'. You can assume that the length of the string is a multiple of 3 and not great than 70.

Output

For each case, print the case number and the total number of valid configurations.

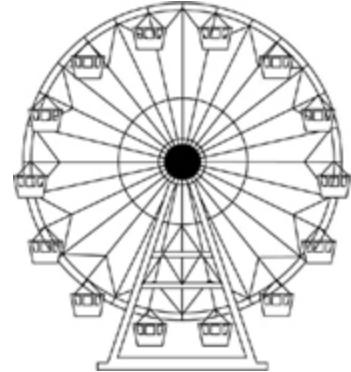
Sample Input	Output for Sample Input
3 RBBRB BRBRBB BBBBBBBB	Case 1: 2 Case 2: 2 Case 3: 12

1303 - Ferris Wheel

As we all know that not only children but Ferris-Wheel (FW) is favorite to all ages. That's why the LightOJ Park has made a special FW for the programmers. And unlike other Ferris-Wheels, this FW has m seats and every seat is different and very special. Since its circular, the seats are numbered from 1 to m in anticlockwise order, so seat 2 is adjacent to seat 1 and seat 3, seat 1 is adjacent to seat 2 and seat m .

One day n programmers (identified by 1 to n) came and each of them wanted to ride in the FW. Since every seat is special, everyone wants to taste the ride of every seat. So, they made the following algorithm:

1. They will form a queue from 1 (**front**) to n (**back**). As they want to enjoy the ride, they want to sit alone in a seat, because two (or more) may start problem-solving in the ride.
2. The FW moves in clockwise order. Initially the seat 1 is in bottom and after 5 seconds, it starts. And when it starts, in each 5 second interval the next seat comes to bottom. So, the order of the seats in bottom are seat 1 (at time 5), seat 2 (at time 10), ..., seat m (at time $5*m$), seat 1 (at time $5*m+5$), ... and in this interval, a person sits in that seat or gets out (if he is in the seat) or both (one gets out and another gets in). Assume that these kinds of movements take quite a small time and thus that can be ignored.
3. When a programmer gets out from one seat (he just rode in that seat), then if he has ridden in all the seats, he will leave, otherwise he will join in the back of the queue.
4. When a seat comes into bottom, if all the programmers in the queue have ridden in the seat, nothing happens. Otherwise the first person (from front) in the queue who hasn't ridden in the seat sits in that seat and other programmers keep standing. For example, the current queue is 5, 2, 3, 1 and a seat is in the bottom which has been already ridden by 5 and 2 but 3 hasn't; so, 3 will sit in that seat leaving the queue: 5, 2, 1.



Now you are the $(n+1)^{\text{th}}$ programmer and you are unlucky, because you are assigned a task, and the task is to find the minimum time when you are sure that everyone has ridden in all the seats.

Input

Input starts with an integer T (≤ 400), denoting the number of test cases.

Each case starts with a line containing two integers n ($1 \leq n \leq 20$) and m ($2 \leq m \leq 20$).

Output

For each case, print the case number and the time.

Sample Input	Output for Sample Input
2 2 3 3 2	Case 1: 65 Case 2: 40

1304 - The Best Contest Site Ever

We are planning to arrange a National Collegiate Programming Contest in Bangladesh. But one of the problems is that members from different teams can discuss with each other if the security is not tight. And this fact sometimes affects the contest rank list. Even if we seize the electronic devices from the team members; there are other ways to communicate. So, our target is to make a site that is fair. The site can be modeled as an **M x N** grid, where **M** is the number of rows, and **N** is the number of columns. Three kinds of cells are there 1) a blank space, 2) a wall, 3) a reserved space. Teams can only be placed in blank spaces, but at most one team can be placed in one blank space. Team members are not allowed to move from their current place, but they can see through the blank or reserved spaces, but not through walls. And they can see vertically or horizontally. To be more specific two members can communicate if they are in the same row (or column) and there is no **wall** between them in that row (or column) (like a chess rook). As we have already discussed that we don't want members of different teams to communicate, we want to place teams in the grid such that there is no way for the members of two different teams to communicate with each other. So, we are assigning this task to you, providing you the information of the grid, your task is to find the **maximum** number of teams we can place in the grid, so that we can put extra efforts on the problem set. You have to find a valid placement of teams. As there can be many solutions with maximum number of teams, any valid one will do.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers **M ($1 \leq M \leq 100$)** and **N ($1 \leq N \leq 100$)**. Each of the next **M** lines contains **N** characters denoting the grid. There will be three kinds of characters. A **'.'** denotes a blank space, a **'W'** denotes a wall and an **'R'** denotes a reserved space.

Output

For each case, print the case number and the total number of teams in a line. Then print the grid using the same format as in input and report the team cells with **'T'**.

Sample Input	Output for Sample Input
2 3 5 .RRR. WWWWW .RRR. 3 5 . .RR. .W.RW .RRR.	Case 1: 2 TRRR. WWWWW TRRR. Case 2: 4 .TRR. TWTRW .RRRT

Note

This is a special judge problem, wrong output format may cause 'wrong answer'.

1305 – Area of a Parallelogram

A parallelogram is a quadrilateral with two pairs of parallel sides. See the picture below:

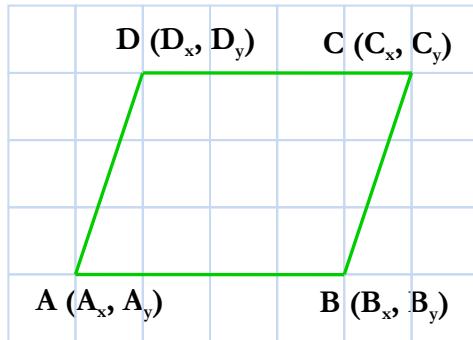


Fig: a parallelogram

Now you are given the co ordinates of **A**, **B** and **C**, you have to find the coordinates of **D** and the area of the parallelogram. The orientation of **ABCD** should be same as in the picture.

Input

Input starts with an integer **T (≤ 1000)**, denoting the number of test cases.

Each case starts with a line containing six integers $A_x, A_y, B_x, B_y, C_x, C_y$ where (A_x, A_y) denotes the coordinate of **A**, (B_x, B_y) denotes the coordinate of **B** and (C_x, C_y) denotes the coordinate of **C**. Value of any coordinate lies in the range **[-1000, 1000]**. And you can assume that **A**, **B** and **C** will not be collinear.

Output

For each case, print the case number and three integers where the first two should be the coordinate of **D** and the third one should be the area of the parallelogram.

Sample Input	Output for Sample Input
3 0 0 10 0 10 10 0 0 10 0 10 -20 -12 -10 21 21 1 40	Case 1: 0 10 100 Case 2: 0 -20 200 Case 3: -32 9 1247

1306 - Solutions to an Equation

You have to find the number of solutions of the following equation:

$$Ax + By + C = 0$$

Where **A**, **B**, **C**, **x**, **y** are integers and $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$.

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case starts with a line containing seven integers **A**, **B**, **C**, **x₁**, **x₂**, **y₁**, **y₂** ($x_1 \leq x_2$, $y_1 \leq y_2$). The value of each integer will lie in the range $[-10^8, 10^8]$.

Output

For each case, print the case number and the total number of solutions.

Sample Input	Output for Sample Input
5 1 1 -5 -5 10 2 4 -10 -8 80 -100 100 -90 90 2 3 -4 1 7 0 8 -2 -3 6 -2 5 -10 5 1 8 -32 0 0 1 10	Case 1: 3 Case 2: 37 Case 3: 1 Case 4: 2 Case 5: 1

1307 – Counting Triangles

You are given N sticks having distinct lengths; you have to form some triangles using the sticks. A triangle is valid if its area is positive. Your task is to find the number of ways you can form a valid triangle using the sticks.

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case starts with a line containing an integer N ($3 \leq N \leq 2000$). The next line contains N integers denoting the lengths of the sticks. You can assume that the lengths are distinct and each length lies in the range $[1, 10^9]$.

Output

For each case, print the case number and the total number of ways a valid triangle can be formed.

Sample Input	Output for Sample Input
3 5 3 12 5 4 9 6 1 2 3 4 5 6 4 100 211 212 121	Case 1: 3 Case 2: 7 Case 3: 4

1308 - Ant Network

Ants made a large underground network for their communication, consisting of **n** junctions, and junctions are connected by **m** underground tunnels. As there are animals/insects that can attack them outside the ground, they made the full network under the ground such that it becomes a safe hideout for them.

The junctions and tunnels are strong but if there is any kind of natural disasters like earthquakes or tornadoes, there is a chance that a junction may collapse. That's why they want to built some escape shafts in junctions (at most one shaft in one junction) that lead them to the surface. Now they want to build minimum number of shafts such that if any of the junctions (only one) collapses, ants that survive the collapse, still have a path to the surface. Now your task is to find the minimum number of shafts, and the number of ways the minimum shafts can be built.

Input

Input starts with an integer **T** (≤ 30), denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers: **n** ($2 \leq n \leq 10000$) and **m** ($0 \leq m \leq 20000$). Each of the next **m** lines contains two integers **u v** ($0 \leq u, v < n, u \neq v$) meaning that there is a bidirectional tunnel between junction **u** and **v**. The input follows the above constraints. And no tunnel is reported more than once. All junctions are connected.

Output

For each case, print the case number, the minimum number of escape shafts and the number of ways they can build the minimum shafts modulo 2^{64} .

Sample Input	Output for Sample Input
2 6 6 0 3 0 1 1 4 4 2 2 5 5 0 5 4 2 1 1 3 0 4 4 1	Case 1: 2 4 Case 2: 3 1

Note

Dataset is huge, use faster I/O methods.

1309 – Children's Math

My youngest sister Rimi came to me with a sad expression in her face. I was thinking what might be the reason, and she revealed it after a while, and that is she was given a math homework and the homework is to solve a linear equation with one variable. Here is the homework:

$$2 * x + 3 * 2 - (7 + 5) = x + 25$$

From this equation, she has to find the value of x . That's why she asked me to write a program to solve any type of linear equations with one variable. And since she is only a child, it's clear that she will not give any expression that may lead to a non linear sub expressions like $(x*x)$, or even she will not give any inputs that lead to a sub expression like $(x*x - x*x + x)$. And she is not familiar with unary operators, so, she'll use binary operators only, but not division (or modulo) operators. But she may use brackets. So, before start coding, I wrote the following grammar:

Equation	=>	Expression '=' Expression
Expression	=>	Term Expression '+' Term Expression '-' Term
Term	=>	Factor Factor '*' Factor
Factor	=>	Number 'x' '(' Expression ')'
Number	=>	Digit Digit Number
Digit	=>	'0' '1' ... '9'

Table 1: Grammar in EBNF

Though the grammar can produce non linear sub expressions, but I am sure that she will not give any such inputs. And numbers will not contain leading zeroes. But when I just started coding, the power was gone due to load shedding, that's why I am asking your help.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing a string E ($1 \leq \text{length}(E) \leq 100$) denoting an equation. The equation follows the above rules. No spaces will be given in the string. And the input is given such that, in any calculation, the absolute value of the result (or operand) will not exceed 10^9 .

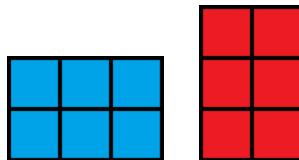
Output

For each case, print the case number first. Then if x has infinitely many solutions, print "**infinitely many solutions**". Print "**no solution**" if there is no such solution. If x has only one solution, print x in p/q form if x is not an integer (p and q should be relatively prime, $q > 0$), otherwise print x .

Sample Input	Output for Sample Input
2 $x^5 + 31 = 7 - (2 * 20) + 53$ $x^2 + (3 * x - 7) = 2 * x + (x - 5) + (2 * x - 2)$	Case 1: -11/5 Case 2: infinitely many solutions

1310 - Tiles (III)

There is an $M \times N$ board, two types of tiles are available, and each of them is infinitely many, you have to place maximum number of non-overlapping tiles in the board. The tiles are given below:



You **cannot** rotate or flip any tile. Some cells in the board may be broken; you can't place any part of a tile in the broken cells.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers: $M N$ ($2 \leq M \leq 8, 2 \leq N \leq 100$). Each of the next M lines contains N characters forming the board. There are two types of characters. A '.' means the cell is **not** broken; a '#' means the cell is **broken**.

Output

For each case, print the case number and maximum number of tiles that can be placed in the board.

Sample Input	Output for Sample Input
3 2 3 2 3 ...# ... 5 6 . # # # ...	Case 1: 1 Case 2: 0 Case 3: 3

1311 - Unlucky Bird

A bird was flying on a train line, singing and passing lazy times. After a while it saw a train coming from its behind, so, it speeded up a bit, but remained calm. After a while it saw another train coming towards it from the front side. The bird remained calm thinking that the train coming towards him would be in another line. But after a while it realized that both the trains were on the same line!

So, the stupid brave bird made a plan to stop the accident. So it flew towards the train which was coming from the front side, and after touching the train the bird turned back immediately, and flew until it could touch another train. And after that it turned back, and continued this procedure. The birds' intention was to signal the drivers such that they could stop the train.

When the trains were d meter way, the drivers realized the abnormal behavior of the strange bird, and saw the opposite trains, and both drivers braked hard! But alas! They were able to stop the collision, but they managed to stop in front of each other leaving no distance. And the brave bird was dead in the middle of the trains. Thousand lives saved, but none remembered the bird.

For simplicity we denote the train (that was behind the bird) as the left train and the other one as the right train. The left train had velocity $v_1 \text{ m/s}$ (meter per second) and the right train had velocity $v_2 \text{ m/s}$ and they saw each other when they were d meter away. The driver in the left train made a deceleration of $a_1 \text{ m/s}^2$ and the driver in the right train made a deceleration of $a_2 \text{ m/s}^2$. And the trains just avoided collision. That means they just stopped when their distance was 0 meter. The bird had constant velocity of $v_3 \text{ m/s}$. And assume that the bird can turn immediately and can keep its constant velocity. When the trains were d meter away, the bird was somewhere between the trains. Your task is to find the distance covered by the brave bird (from this moment) in meters before sacrificing its life for thousand lives.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing five positive real numbers: $v_1 \ v_2 \ v_3 \ a_1 \ a_2$ ($v_1 < v_3, v_2 < v_3$). No real number will be greater than 1000 . And no number contains more than three digits after the decimal point.

Output

For each case, print the case number, d and the distance covered by the bird. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
1 0.5 1.0 2 0.25 0.5	Case 1: 1.50000000 4.0

1312 – Corrupted Tax Department

There is a strange country, for simplicity we name the country as 'You Know Which', you will know the country after reading the following. The Income Tax department of the country is so corrupted that people used to say that after paying the tax, they are stolen by 'You Know Who'. So, the country is considered as an underdeveloped but leading corrupted countries for a few decades (may be we will see a century or even millennium!). So, people are dissatisfied with the Govt. but still all the taxes paid by general people are taken by 'You Know Who'.

So, some of the talented ACM solvers have realized the fact and they took some initiatives to stop the corruption in the income tax department, and want to save the department from the mysterious 'You Know Who'. In the income tax department, there is an officer who is the head of the department. Every other officer is either subordinate of him or any other officer. But any officer is direct or indirect subordinate of the head. An officer is a subordinate of exactly one other officer.

As the ACM-ers don't know which officers (or the head) are corrupted, so, they planned that every officer or the head should work with his direct subordinate and these two should make a group. The fact is that since they are not in same level, it would be tough for them to join 'You Know Who'. It may be possible that some members cannot form a group, so the ACM-ers want to form maximum number of groups. And they also want to find the number of ways they can form maximum groups.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 10000$)**, where **n** denotes the number of officers (including the head). The head is identified by **1** and others are identified from **2** to **n**. Each of the next **n** lines contains the id of the person, the number of subordinates of this person, and the list of ids of the subordinates of this person separated by spaces. In the list no id is given more than once, and you can assume that the given input follows the restrictions described above.

Output

For each case, print the case number, the maximum number of groups and the number of ways to make maximum groups modulo **10007**.

Sample Input	Output for Sample Input
1 5 1 2 2 4 4 1 5 2 1 3 3 0 5 0	Case 1: 2 3

Note

Dataset is huge, use faster I/O methods.

1313 - Protect the Mines

You are a rich man, and recently you bought a place which contains some mines of rare metals like gold, platinum etc. As the metals are rare, their cost is also so high. So, you need to protect them from thieves. But things are not as easy as it looks.

So, you called a bunch of engineers to make some wired fences around the mines. As they were just engineers (not problem solvers!), they drilled some holes in random places. The idea was to put some pillars on the drilled holes and after that some wires should be set in the pillars, and finally electrifying the wires would be the completion. And of course, each mine should be surrounded by a fence. Wires should be placed between two pillars, in straight lines. The engineers drilled the holes in positions such that some of the mines might not be covered by any fences.

However, your plan is that, you can put some guards in mines that are not surrounded by any fence. To guard a single mine, it would cost **G** dollars, and a pillar cost is **P** dollars. As you are a rich man, the costs of wires are too small that they can be ignored. So, you want to put guards in some mines and surround some other mines by fences, but you want to find the optimal cost to protect all the mines. The fences may or may not be convex.

In bird's eye view, we can get the following figure (fig 1) for sample 1. There are seven pre-drilled holes (marked as circles), and six mine positions (marked as squares). The straight lines show the wires and the closed regions form the fences. The figure shows one of the optimal solutions.

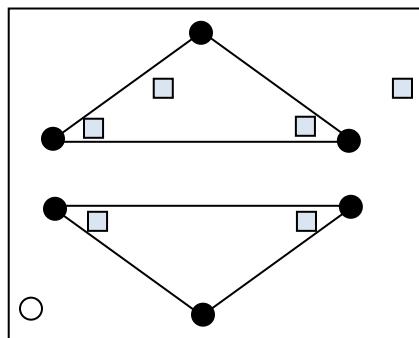


Fig 1

So, you have to find the minimum cost for executing your plan.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing four integers **N** ($3 \leq N \leq 100$), **M** ($1 \leq M \leq 100$), **G** ($1000 \leq G \leq 2000$) and **P** ($100 \leq P \leq 200$), **N** denotes the number of pre-drilled holes, and **M** denotes the number of mines. This line is followed by **N** lines that describe the positions of the holes, and then by **M** lines that describe the positions of the mines. All positions are given as pairs of integers **x y** on one line ($0 \leq x, y \leq 1000$). You can assume that no two positions (of holes and mines) coincide and that no three positions are collinear.

Output

For each case, print the case number and the minimum cost to protect all the mines.

Sample Input	Output for Sample Input
2 7 6 1000 100 0 0 20 0 1 10 39 10 1 20 39 20 20 30 3 9 37 9 3 21 37 21 18 24 50 24 4 3 1500 100 0 0 0 10 10 0 10 10 5 4 4 1 8 6	Case 1: 1600 Case 2: 300

1314 – Names for Babies

Long time ago, there was a strange kingdom. Peoples of different religions, different cultures used to live there. But as they were different, their names were also different. So, in schools, offices, it was quite tough to call someone using his/her name, because some names were too hard to be pronounced by persons from different culture.

So, the king made a plan. He took a string **S** and two integers **p** and **q** and made a rule that names of the babies should be a substring of **S**, and the length should be between **p** and **q** (inclusive).

Now you are given **S**, **p** and **q** you have to find the number of distinct names that can be made.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing a string **S**. The length of **S** will be between **2** and **10000** (inclusive) and **S** contains lowercase English letters only. The next line contains two integer **p** and **q** ($1 \leq p \leq q \leq \text{length}(S)$).

Output

For each case, print the case number and the number of distinct names that can be made.

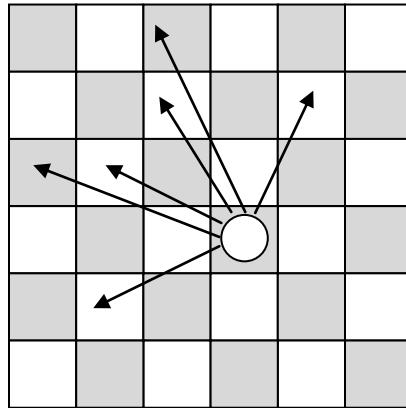
Sample Input	Output for Sample Input
1 abcdef 2 5	Case 1: 14

Note

This problem was used in [contest 12](#). But the length of **S** was between **2** and **100**.

1315 - Game of Hyper Knights

A Hyper Knight is like a chess knight except it has some special moves that a regular knight cannot do. Alice and Bob are playing this game (you may wonder why they always play these games!). As always, they both alternate turns, play optimally and Alice starts first. For this game, there are 6 valid moves for a hyper knight, and they are shown in the following figure (circle shows the knight).



They are playing the game in an infinite chessboard where the upper left cell is (0, 0), the cell right to (0, 0) is (0, 1). There are some hyper knights in the board initially and in each turn a player selects a knight and gives a valid knight move as given. And the player who cannot make a valid move loses. Multiple knights can go to the same cell, but exactly one knight should be moved in each turn.

Now you are given the initial knight positions in the board, you have to find the winner of the game.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 1000$)** where **n** denotes the number of hyper knights. Each of the next **n** lines contains two integers **x y ($0 \leq x, y < 500$)** denoting the position of a knight.

Output

For each case, print the case number and the name of the winning player.

Sample Input	Output for Sample Input
2 1 1 0 2 2 5 3 5	Case 1: Bob Case 2: Alice

1316 – A Wedding Party

We all know that we have a big and exciting wedding party ahead. So we made a plan to go to buy a gift for the wedding party. We all gathered at a place and we were just about to buy the gift. Unfortunately, we find that we have a "Team Practice Contest" ahead. Now before going to the contest we have to buy the gift. As time is too short we will try to buy the gift on the way to the contest. We will try to visit as many shops as possible. The city map is represented by a graph with **N** nodes and **M** edges. **N** nodes represent the **N** junctions and **M** edges represent the **M** unidirectional roads connecting the cities. Every road has a cost which represents the required time to use the road. The contest is running at junction **N-1** and we will start our journey at junction **0**. And there are exactly **S** shops located at different junctions.

Now given the location of the shops you have to find the route from junction **0** to junction **N-1** which will visit maximum number of shops with minimum time (first maximize the number of shops then minimize the time to visit them). We can visit a junction more than once.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

Each case begins with three non negative integers **N** ($2 \leq N \leq 500$), **M** ($1 \leq M \leq 10000$) **S** ($0 \leq S \leq 15$). Next line contains **S** integers denoting the shop locations. Each of the next **M** lines contains three integers **u, v, w** ($0 < u, v < N, u \neq v, 1 \leq w \leq 100$) denoting a road from **u** to **v** with cost **w**.

Output

For each case of input you have to print the case number and two integers representing maximum number of shops we can visit in the way and the minimum time required to reach junction **N-1** after visiting maximum number of shops. If we cannot attend the contest, print "**Impossible**". See samples for more details.

Sample Input	Output for Sample Input
2 4 4 4 0 1 2 3 0 1 10 1 3 30 0 2 30 2 3 5 4 4 4 0 1 2 3 0 1 10 3 1 30 0 2 30 3 2 5	Case 1: 3 35 Case 2: Impossible

1317 – Throwing Balls into the Baskets

You probably have played the game "**Throwing Balls into the Basket**". It is a simple game. You have to throw a ball into a basket from a certain distance. One day we (the AIUB ACMMER) were playing the game. But it was slightly different from the main game. In our game we were **N** people trying to throw balls into **M** identical Baskets. At each turn we all were selecting a basket and trying to throw a ball into it. After the game we saw exactly **S** balls were successful. Now you will be given the value of **N** and **M**. For each player probability of throwing a ball into any basket successfully is **P**. Assume that there are infinitely many balls and the probability of choosing a basket by any player is **1/M**. If multiple people choose a common basket and throw their ball, you can assume that their balls will not conflict, and the probability remains same for getting inside a basket. You have to find the expected number of balls entered into the baskets after **K** turns.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing three integers **N** ($1 \leq N \leq 16$), **M** ($1 \leq M \leq 100$) and **K** ($0 \leq K \leq 100$) and a real number **P** ($0 \leq P \leq 1$). **P** contains at most three places after the decimal point.

Output

For each case, print the case number and the expected number of balls. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 1 1 1 0.5 1 1 2 0.5	Case 1: 0.5 Case 2: 1.000000

1318 – Strange Game

In a country named "Ajob Desh", people play a game called "Ajob Game" (or strange game). This game is actually a game of words. The rules for the game are as follows:

1. It's an \mathbf{N} player game and players are numbered from 1 to \mathbf{N} . And the players alternate turns in a circular way. Player 1 starts first. The next turn is for player 2 , then player 3 and so on. After the turn for the \mathbf{N}^{th} player, player 1 gets his turn again and the same procedure is continued.
2. In each turn a player has to propose a pair of words. Each of the words should have length \mathbf{L} , and the words should differ in exactly \mathbf{M} positions. As their language has \mathbf{K} alphabetical symbols, a word is a collection of symbols from these \mathbf{K} alphabets.
3. The pair of words proposed by a player should differ in exactly \mathbf{M} positions, it means that there should be exactly \mathbf{M} positions where the two words have different symbols, and in other positions they have same symbols. For example, ' abc ' and ' abd ' differ in exactly 1 position, ' abc ' and ' aca ' differ in exactly 2 positions, ' abc ' and ' cab ' differ in exactly 3 positions.
4. In each turn a player has to propose a new pair of words. Two pairs are different if at least one word is different. Note that here pair refers to unordered pair. Let \mathbf{A} , \mathbf{B} , \mathbf{C} be three different words, then (\mathbf{A}, \mathbf{B}) and (\mathbf{B}, \mathbf{A}) are same, but (\mathbf{A}, \mathbf{C}) and (\mathbf{A}, \mathbf{B}) are different. For example, if a player already proposed $\{\text{abc}, \text{def}\}$, then none can propose $\{\text{abc}, \text{def}\}$ or $\{\text{def}, \text{abc}\}$. But a player can propose $\{\text{abc}, \text{fed}\}$ or $\{\text{abc}, \text{abc}\}$ or $\{\text{pqc}, \text{abc}\}$ etc.
5. If a player fails to propose a new pair of words, he is treated as the loser of the game. And the game ends.

Let $\mathbf{N} = 2$, $\mathbf{K} = 2$, $\mathbf{L} = 2$, $\mathbf{M} = 1$ and the alphabet is $\{\text{ab}\}$. All the words of length 2 are: $\{\text{aa}, \text{ab}, \text{ba}, \text{bb}\}$. Player 1 chooses pair $\{\text{aa}, \text{ab}\}$ (differs in 1 position as $\mathbf{M} = 1$) then player 2 chooses pair $\{\text{ab}, \text{bb}\}$. After that player 1 chooses $\{\text{aa}, \text{ba}\}$ then player 2 chooses $\{\text{bb}, \text{ba}\}$. And then there is no pair left for player 1 , and so, player 1 will lose.

Now this game is played by \mathbf{N} players who know this game very well thus they play optimally. You are given \mathbf{N} , \mathbf{K} , \mathbf{L} and \mathbf{M} ; you have to find the loosing player.

Input

Input starts with an integer \mathbf{T} (≤ 200), denoting the number of test cases.

Each case starts with a line containing four integers \mathbf{N} ($2 \leq \mathbf{N} \leq 10000$), \mathbf{K} ($1 \leq \mathbf{K} \leq 10^9$), \mathbf{L} ($1 \leq \mathbf{L} \leq 10^5$) and \mathbf{M} ($0 \leq \mathbf{M} \leq \mathbf{L}$).

Output

For each case, print the case number and the player who loses the game.

Sample Input	Output for Sample Input
5	Case 1: 1
2 2 2 1	Case 2: 1
3 4 3 3	Case 3: 5
9 26 8 5	Case 4: 3
10 2 2 2	Case 5: 10
100 3 2 0	

1319 - Monkey Tradition

In 'MonkeyLand', there is a traditional game called "Bamboo Climbing". The rules of the game are as follows:

- 1) There are N monkeys who play this game and there are N bamboos of equal heights. Let the height be L meters.
- 2) Each monkey stands in front of a bamboo and every monkey is assigned a different bamboo.
- 3) When the whistle is blown, the monkeys start climbing the bamboos and they are not allowed to jump to a different bamboo throughout the game.
- 4) Since they are monkeys, they usually climb by jumping. And in each jump, the i^{th} monkey can jump exactly p_i meters (p_i is a prime). After a while when a monkey finds that he cannot jump because one more jump may get him out of the bamboo, he reports the remaining length r_i that he is not able to cover.
- 5) And before the game, each monkey is assigned a distinct p_i .
- 6) The monkey, who has the lowest r_i , wins.

Now, the organizers have found all the information of the game last year, but unluckily they haven't found the height of the bamboo. To be more exact, they know N , all p_i and corresponding r_i , but not L . So, you came forward and found the task challenging and so, you want to find L , from the given information.

Input

Input starts with an integer T (≤ 10000), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 12$). Each of the next n lines contains two integers p_i ($1 < p_i < 40$, p_i is a prime) and r_i ($0 < r_i < p_i$). All p_i will be distinct.

Output

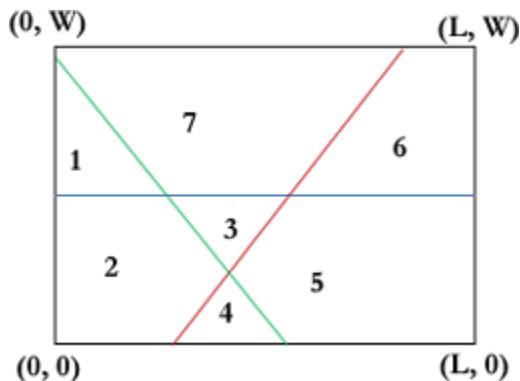
For each case, print the case number and the minimum possible value of L that satisfies the above conditions. If there is no solution, print '**Impossible**'.

Sample Input	Output for Sample Input
2 3 5 4 7 6 11 3 4 2 1 3 2 5 3 7 1	Case 1: 69 Case 2: 113

1320 – Farmers' Tale

You may have heard the story. There was a farmer and he had a big rectangular land. When he grew older he divided this land to his sons. His sons also divided their shares to their sons when they grew older. And this way the poor land was getting divided and divided. And this way when the population increases, the area of total agricultural lands decrease.

In this problem you have to do a similar task. You are given a rectangular land in 2D space, its height L and width W . The lower left corner of the rectangle is $(0, 0)$ and the upper right corner is (L, W) . There will be some lines that connect two different sides of the rectangle. You have to find the number of different parts of the rectangular area after drawing the lines. In the picture, there are three lines. And there are seven different parts in the rectangle.



Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing three integers n ($0 \leq n \leq 100$), L , W ($5 \leq L, W \leq 100$). Each of the next n lines contains four integers x_1 , y_1 , x_2 , y_2 denoting a line in the rectangle. You can assume that (x_1, y_1) and (x_2, y_2) are on the boundary of the rectangle in different sides. No line is given more than once.

Output

For each case, print the case number and the number of different parts.

Sample Input	Output for Sample Input
2 3 100 100 0 50 100 50 50 0 0 93 30 0 80 100 0 10 10	Case 1: 7 Case 2: 1

Note

Better to use integer calculations for this problem. Precision may cause problems.

1321 - Sending Packets

Alice and Bob are trying to communicate through the internet. Just assume that there are **N** routers in the internet and they are numbered from **0** to **N-1**. Alice is directly connected to router **0** and Bob is directly connected to router **N-1**. Alice initiates the connection and she wants to send **S** KB of data to Bob. Data can go to the (**N-1**)th router from the **0th** router either directly or via some intermediate routers. There are some bidirectional links between some routers.

The links between the routers are not necessarily 100% perfect. So, for each link, a probability **p_i** is given. That means if **u** and **v** are two routers and if their underlying link has probability **p_i**, it means that if data is sent from **u** to **v**, the probability of successfully getting the data in **v** is **p_i** and vice versa. If multiple links are used the probability of getting the data in destination is the multiplication of the probabilities of the links that have been used.

Assume that it takes exactly **K** seconds for a packet to reach Bob's router from Alice's router (independent on the number of links) if it's successful. And when the data is successfully received in Bob's router, it immediately sends an acknowledgement to Alice's router and the acknowledgement always reaches her router exactly in **K** seconds (it never disappears).

Alice's router used the following algorithm for the data communication.

- 1) At time 0, the first KB of data is chosen to be sent.
- 2) It establishes a path (it takes no time) to the destination router and sends the data in this route.
- 3) It waits for exactly **2K** seconds.
 - a. If it gets the acknowledgement of the current data in this interval
 - i. If **S** KB of data are sent, then step 4 is followed.
 - ii. Otherwise, it takes 1 KB of the next data, and then step 2 is followed.
 - b. Otherwise it resends the current 1 KB of data and then step 2 is followed.
- 4) All the data are sent, so it reports Alice.

Assume that the probabilities of the links are static and independent. That means it doesn't depend on the result of the previously sent data. Now your task is to choose some routes through the routers such that data can be sent in these routes and the expected time to send all the data to the destination routes is minimized. You only have to report the minimum expected time.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing four integers **N** ($2 \leq N \leq 100$), **M** ($1 \leq M$), **S** ($1 \leq S \leq 10^9$) and **K** ($1 \leq K \leq 20$), where **M** denotes the number of bidirectional links. Each of the next **M** lines contains three integers **u_i v_i p_i**, meaning that there is a link between router **u_i** and **v_i** the probability for a successful message transfer in this link is **p_i%** ($0 \leq u_i, v_i < N, u_i \neq v_i, 0 < p_i \leq 100$). There will be at most one link between two routers.

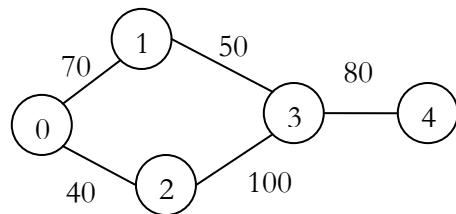
Output

For each case, print the case number and the minimum possible expected time to send all the data. Errors less than 10^{-3} will be ignored. You can assume that at least one valid route between them always exists. And the result will be less than 10^{13} .

Sample Input	Output for Sample Input
2 5 5 1 10 0 1 70 0 2 40 2 3 100 1 3 50 4 3 80 2 1 30 2 0 1 80	Case 1: 62.5000000000 Case 2: 150

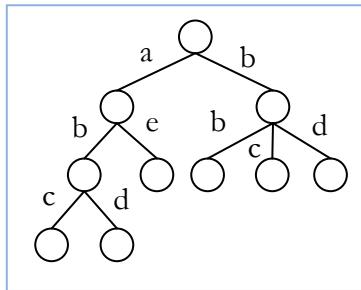
Note

For sample 1, we get the following picture. We send the data through 0 - 2 - 3 - 4.

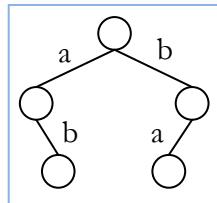


1322 - Worst Case Trie

In Computer Science Trie or prefix tree is a data structure which is usually used to store some strings or some numbers. Unlike binary trees, edges contain characters. And a node actually represents a string which is found by taking the characters from the edges, in the path from root to leaf. For example, for {abc, ae, bd, bb, bc, abd} we get the following trie:



Now you are given a set of strings and each string uses one of the **K** character symbols, and in any string (from the set) a symbol occurs at most once. Your task is to find the number of nodes required if we make a trie with the strings, using the procedure described above. As you don't know the size of the set, your task is to find the worst case result. For example, if you have 2 character symbols, then you need 5 nodes in worst case as in the following trie (let the symbols be {a, b}):



Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case starts with a line containing an integer **K** ($1 \leq K \leq 10^8$).

Output

For each case, print the case number and the total number of nodes required in worst case. The result can be big, print the least 4 significant digits if result has 4 or more digits, otherwise, print the result. [The least 4 significant digits of 123456789 is 6789].

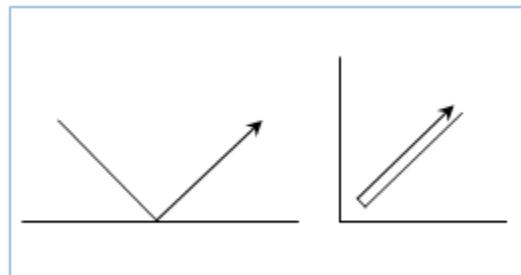
Sample Input	Output for Sample Input
3	Case 1: 2
1	Case 2: 5
2	Case 3: 16
3	

1323 - Billiard Balls

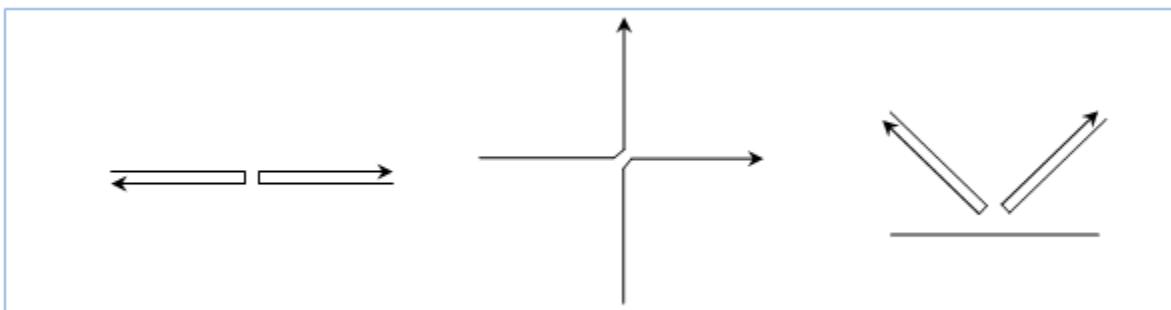
You are given a rectangular billiard board, L and W be the length and width of the board respectively. Unlike other billiard boards it doesn't have any pockets. So, the balls move freely in the board. Assume that initially some balls are in the board and all of them are moving diagonally. Their velocities are same but their direction may be different. When one or more balls bounce off, their position changes but their velocity remains same.

You are given the initial positions of the balls and their directions; your task is to find the position of the balls after K seconds. The board is placed in the 2D plane such that the boundaries are $(0, 0)$, $(L, 0)$, (L, W) and $(0, W)$. The positions of balls are given as 2D co-ordinates and they all lie inside the board. And the directions are given as one of the following $\{NE, SE, SW, NW\}$, N, E, S and W denote North, East, South and West respectively. NE means North-East so both x and y are increasing. The balls are so small that their radii can be said to be 0. In each second, the balls advance one unit in their direction. Here one unit doesn't mean Euclidean one unit. For example, if the current position of a ball is (x, y) and its direction is NW then in the next second its position will be $(x-1, y+1)$.

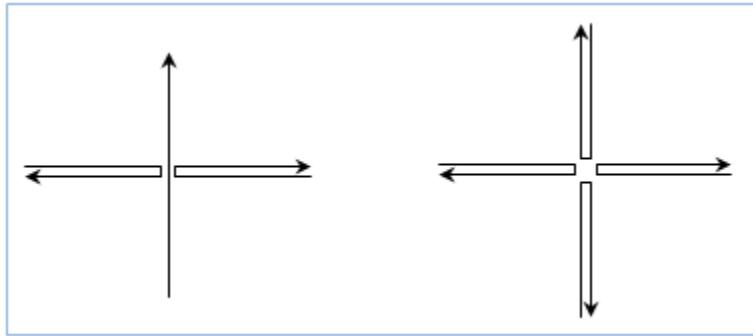
When two or more balls bounce off, they may change their directions as shown in the pictures. You can rotate the pictures to get all possible results. Remember that the balls may bounce at non-integer points.



Bouncing on walls



Bounce result between two balls



Bounce result amongst 3 balls and 4 balls

Input

Input starts with an integer **T (≤ 25)**, denoting the number of test cases.

Each case starts with a line containing four integers **L, W ($5 \leq W \leq L \leq 10^8$)**, **n ($1 \leq n \leq 1000$)** and **K ($1 \leq K \leq 10^8$)** where **n** denotes the number of balls in the board. Each of the next **n** lines contains two integers **x y** and a string **d**, where **(x, y) ($0 \leq x \leq L, 0 \leq y \leq W$)** denotes the co-ordinate of the ball and **d** can be one of {NE, SE, SW, NW} which denotes the direction of the ball respectively. You can safely assume that the balls are placed in different positions initially.

Output

For each case, print the case number in a single line. Then print the position of the balls. Sort the positions first by their **x** co-ordinate, and order the ones which have same **x** coordinate by their **y** coordinates all in ascending order.

Sample Input	Output for Sample Input
<pre> 2 10 5 6 1 1 4 NW 1 1 SW 9 1 SE 8 3 NE 9 4 NE 7 4 NE 10 5 6 4 1 4 NW 1 1 SW 9 1 SE 8 3 NE 9 4 NE 7 4 NE </pre>	<pre> Case 1: 0 0 0 5 8 5 9 4 10 0 10 5 Case 2: 3 2 3 3 7 2 7 3 8 3 9 2 </pre>

1324 – Equivalent Boolean Expressions

In this problem you are given some Boolean expressions, you can evaluate them and find whether they return true or false. A Boolean expression consists of three basic operators, and ($\&$), or ($|$) or not ($!$). ' $\&$ ' and ' $|$ ' are binary operators, where ' $!$ ' is a unary operator. ' $\&$ ' returns true if and only if both of its operands are true, ' $|$ ' returns true if any of its operands is true. ' $!$ ' negates the operand, that means it makes true to false and vice versa. The grammar is given below:

Expression	=>	Term Expression ' $ $ ' Term
Term	=>	Factor Term ' $\&$ ' Factor
Factor	=>	Sub ' $!$ ' Factor
Sub	=>	Variable '(' Expression ')'
Variable	=>	'a' 'b' ... 'j'

Table 1: Grammar in EBNF

And Boolean variables can be true or false. In this problem, you are given two Boolean expressions, your task is to find whether they are equivalent or not. Two Boolean expressions are said equivalent if they produce same results in all situations. That means if we change the value of the variables, the expressions produce same results.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains two lines. In each line there will be a non-empty string representing a Boolean expression which follows the grammar described above. The strings will not contain any white spaces and the length will be less than 101.

Output

For each case, print the case number and '**Equivalent**' if the expressions are equivalent; or '**Not Equivalent**' if they are not. Be careful about spelling and case sensitivity.

Sample Input	Output for Sample Input
5 a b ! (!a & !b) ! (!a !b) a & b a b c a (b & c) (a !a) (c & !c) ! (b & !b) a (b & c) (a & b) (a & c)	Case 1: Equivalent Case 2: Equivalent Case 3: Not Equivalent Case 4: Equivalent Case 5: Not Equivalent

1325 – Distributing Chocolates

I have bought **n** chocolates for my young cousins. Every chocolate is different. So, in the contest I added the problem that how many ways I can distribute the chocolates to my **K** cousins. I can give more chocolates to some cousins, and may give no chocolate to some. For example, I have three cousins and I bought 2 chocolates **a** and **b**. Then I can distribute them in the following 9 ways:

No.	Cousin 1	Cousin 2	Cousin 3
1	a, b		
2		a, b	
3			a, b
4	a	b	
5	a		b
6		a	b
7	b	a	
8	b		a
9		b	a

Now as the result can be large, I asked for the result modulo **100 000 007** (a prime). But after that I found that this problem is easier than I thought. So, I changed the problem a little bit. I will give you the number of my cousins **K** and the result modulo **100 000 007**. Your task is to find the number of chocolates I have bought. If there are several solutions, you have to find the minimum one.

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case starts with a line containing two integers **K** ($2 \leq K \leq 10^7$) and **result** ($0 \leq \text{result} < 100000007$). You can assume that the input data is valid, that means a solution exists.

Output

For each case, print the case number and the minimum possible number of chocolates I have bought.

Sample Input	Output for Sample Input
2 3 9 2 100	Case 1: 2 Case 2: 23502611

1326 - Race

Disky and Sooma, two of the biggest mega minds of Bangladesh went to a far country. They ate, coded and wandered around, even in their holidays. They passed several months in this way. But everything has an end. A holy person, Munsiji came into their life. Munsiji took them to derby (horse racing). Munsiji enjoyed the race, but as usual Disky and Sooma did their as usual task instead of passing some romantic moments. They were thinking- in how many ways a race can finish! Who knows, maybe this is their romance!

In a race there are **n** horses. You have to output the number of ways the race can finish. Note that, more than one horse may get the same position. For example, 2 horses can finish in 3 ways.

1. Both first
2. horse1 first and horse2 second
3. horse2 first and horse1 second

Input

Input starts with an integer **T (≤ 1000)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 1000$)**.

Output

For each case, print the case number and the number of ways the race can finish. The result can be very large, print the result modulo **10056**.

Sample Input	Output for Sample Input
3 1 2 3	Case 1: 1 Case 2: 3 Case 3: 13

1327 – Help the Winners

The Sell N' Profit shop has recently had a raffle contest with a prize of a dress and matching shoes to some of their regular customers. They have set aside an equal number of dresses and pairs of shoes of different designs for the purpose.

The plan was going quite well until one customer pointed out that not all the dresses and all the pairs of shoes match each other (i.e. you won't look good if you wore that dress and that pair of shoes at the same time - but the same dress or shoes might look good with something else). So, simply giving someone a dress and a pair of shoes at random, risks angering their customers. They have to give each winner a dress and shoes that match. But some pairs of (dress, shoes) are so adorable to girls such that if they are given to any girl, she would be happier than ever and she may become a great customer of the shop. These pairs are called super matching pairs. But, choosing such pair may lead to a situation that some other girls get non-matching (dress, shoes) pairs.

Now you are given **n** dresses and **n** pairs of shoes, a list of what dress matches which shoes and some super matching (dress, shoes) pairs. As the company have lack of fashion sense, they ask you to find how many ways there are to make up **n** sets of dresses and matching shoes, such that either all the girls are happy (got matching dress and shoes), or at least one of the girls got a super matching pair (in this case some girls may get non matching pairs).

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with an integer **n** ($1 \leq n \leq 15$) by itself on a line, denoting the number of dresses and pair of shoes. Each of the next **n** lines contains **n** space separated integers (one of **0**, **1** or **2**). The **jth** integer in **ith** line represents the status of the **ith** dress and **jth** pair of shoes. **0** means that **ith** dress doesn't match **jth** pair of shoes. **1** means they match. **2** means they are super matching.

Output

For each case, print the case number and number of ways you can form **n** sets of dresses and shoes.

Sample Input	Output for Sample Input
2 3 0 1 1 1 1 0 1 0 1 3 1 1 2 2 1 0 1 1 2	Case 1: 2 Case 2: 4

1328 - A Gift from the Setter

Problem setting is somewhat of a cruel task of throwing something at the contestants and having them scratch their head to derive a solution. In this problem, the setter is a bit kind and has decided to gift the contestants an algorithm which they should code and submit. The **C/C++** equivalent code of the algorithm is given below:



```
long long GetDiffSum( int a[], int n )
{
    long long sum = 0;
    int i, j;
    for( i = 0; i < n; i++ )
        for( j = i + 1; j < n; j++ )
            sum += abs( a[i] - a[j] ); // abs means absolute value
    return sum;
}
```

The values of array **a[]** are generated by the following recurrence relation:

$$a[i] = (K * a[i-1] + C) \% 1000007 \text{ for } i > 0$$

where **K**, **C** and **a[0]** are predefined values. In this problem, given the values of **K**, **C**, **n** and **a[0]**, you have find the result of the function

"**long long GetDiffSum(int a[], int n)**"

But the setter soon realizes that the straight forward implementation of the code is not efficient enough and may return the famous "TLE" and that's why he asks you to optimize the code.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains four integers **K**, **C**, **n** and **a[0]**. You can assume that $(1 \leq K, C, a[0] \leq 10^4)$ and $(2 \leq n \leq 10^5)$.

Output

For each case, print the case number and the value returned by the function as stated above.

Sample Input	Output for Sample Input
2 1 1 2 1 10 10 10 5	Case 1: 1 Case 2: 7136758

1329 – Playing Cards

In a regular card set there are 52 cards, each card has two parts, the value and the suit. The values are one of **2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A** and the suit is one of **H, S, D, C**. Cards are represented first by their value and then by their suit. For example, **7H, 2C, JD** etc are some cards.

Now from the regular card set, you removed some of the cards (probably 0). Your task is to find the number of ways you can place the cards in a line such that no two adjacent cards have the same value. You have to use all the cards (of course the remaining cards).

For example, **2H, 3C, 5C** is a valid solution, but **5H, 5C, 7S** is not.

Input

Input starts with an integer **T (≤ 20000)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 52$)** denoting the number of **remaining** cards, followed by **n** remaining cards (a single space precedes every card). Assume that the cards are from the regular set and no card is reported more than once. Also assume that the cards will be represented as stated above.

Output

For each case, print the case number and the number of ways to place the cards in a line such that no two adjacent cards have the same value. Print the result modulo 2^{64} .

Sample Input	Output for Sample Input
5 1 TC 2 TC TS 5 2C AD AC JC JH 4 AC KC QC JC 6 AC AD AS JC JD KD	Case 1: 1 Case 2: 0 Case 3: 48 Case 4: 24 Case 5: 120

1330 – Binary Matrix

A binary matrix is an $m \times n$ matrix consisting of only zeroes and ones. Now you are given m integers, i^{th} integer indicating the summation of the values of cells in i^{th} row. You are also given n integers, j^{th} integer indicating the summation of the values of cells in j^{th} column.

Your task is to generate the binary matrix. As there can be multiple solutions, we want the solution which is lexicographically smallest. To compare two solutions, we first find the cell (topmost, then leftmost) where the solutions differ; then the solution which contains 0 in that cell is lexicographically smaller. So,

$$\begin{array}{ccc} 001 & & 001 \\ 010 & < & 100 \\ 100 & & 010 \end{array}$$

Input

Input starts with an integer T (≤ 125), denoting the number of test cases.

Each case starts with a line containing two integers: m and n ($1 \leq m, n \leq 50$). The next line contains m integers, separated by a single space, denoting the row sums. The next line contains n integers, separated by spaces, denoting the column sum. All the integers will be between **0** and **50** (inclusive).

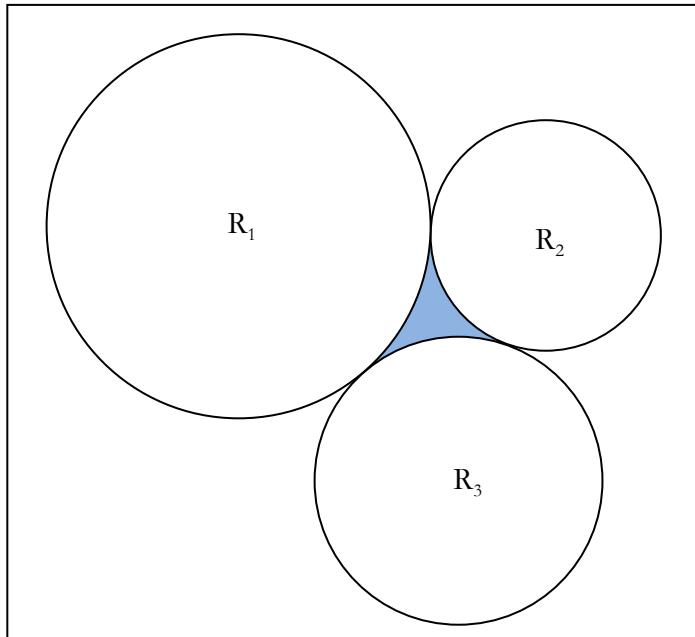
Output

For each case, print the case number first. Then if there is no solution, then print '**impossible**' on the same line. Otherwise, from the next line, print m lines each having n characters denoting the binary matrix as stated above.

Sample Input	Output for Sample Input
5 3 3 1 1 1 1 1 1 3 3 1 1 2 2 2 1 2 3 30 30 30 20 10 2 9 5 5 1 1 2 1 1 1 1 1 1 3 3 1 2 3 3 2 1	Case 1: 001 010 100 Case 2: impossible Case 3: impossible Case 4: 001001111 111110000 Case 5: 100 110 111

1331 – Agent J

Agent **J** is preparing to steal an antique diamond piece from a museum. As it is fully guarded and they are guarding it using high technologies, it's not easy to steal the piece. There are three circular laser scanners in the museum which are the main headache for Agent **J**. The scanners are centered in a certain position, and they keep rotating maintaining a certain radius. And they are placed such that their coverage areas **touch** each other as shown in the picture below:



Here **R**₁, **R**₂ and **R**₃ are the radii of the coverage areas of the three laser scanners. The diamond is placed in the place blue shaded region as in the picture. Now your task is to find the area of this region for Agent **J**, as he needs to know where he should land to steal the diamond.

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case starts with a line containing three real numbers denoting **R**₁, **R**₂ and **R**₃ ($0 < R_1, R_2, R_3 \leq 100$). And no number contains more than two digits after the decimal point.

Output

For each case, print the case number and the area of the place where the diamond piece is located. Error less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3 1.0 1.0 1.0 2 2 2 3 3 3	Case 1: 0.16125448 Case 2: 0.645017923 Case 3: 1.4512903270

1332 – Kings in Chessboard

A regular **King** in a chess board can attack all its adjacent 8 cells (vertical, horizontal or diagonal). Now you are given a **10 x n** chessboard, your task is to place exactly two kings in each column such that no king attacks another. You have to report the number of ways to do that.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 10^9$)**.

Output

For each case, print the case number and the number of ways to place kings in the chessboard such that no king attacks another and every column contains two kings. Print the result modulo 2^{32} .

Sample Input	Output for Sample Input
3	Case 1: 36
1	Case 2: 210
2	Case 3: 1350
3	

1333 – Grid Coloring

You have to color an $M \times N$ two dimensional grid. You will be provided K different colors for this. You will also be provided a list of B blocked cells of this grid. You cannot color these blocked cells.

A cell can be described as (x, y) , which points to the y^{th} cell from the left of the x^{th} row from the top.

While coloring the grid, you have to follow these rules -

1. You have to color each cell which is not blocked.
2. You cannot color a blocked cell.
3. You can choose exactly one color from K given colors to color a cell.
4. No two vertically adjacent cells can have the same color, i.e. cell (x, y) and cell $(x + 1, y)$ shouldn't contain the same color.

You have to calculate the number of ways you can color this grid obeying all the rules provided.

Input

Input starts with an integer T (≤ 600), denoting the number of test cases.

Each test case starts with a line containing four integers $M \ N \ K \ B$ ($1 \leq M, N, K \leq 10^6, 0 \leq B \leq 500$). Each of the next B lines will contain two integers x and y ($1 \leq x \leq M, 1 \leq y \leq N$), the row and column number of a blocked cell. Each of these B lines will contain distinct cells.

Output

For each case, print the case number and the number of ways to color the grid modulo 10^9 .

Sample Input	Output for Sample Input
3 3 3 3 0 3 4 4 2 3 1 3 3 2 2 5 2 1 2 2 2	Case 1: 1728 Case 2: 186624 Case 3: 20

1334 – Genes in DNA

Your friend is a biologist. He has just sequenced a DNA and wants to know about contribution of different genes in that DNA. Both Gene and DNA can be represented by a sequence of letters or strings. Given the sequence of a DNA **D** and a Gene **G**; your friend uses following method to calculate the contribution.

1. Generate a list **P** of proper non-empty prefixes of **G** and another **S** of proper non-empty suffixes of **G** [1]. Additionally let the **L** is list of all strings that is concatenation of a prefix and a suffix. So if **G = ACCT** then **P = A, AC, ACC** and **S = T, CT, CCT** and **L = AT, ACT, ACCT, ACT, ACCT, ACCCT, ACCT, ACCCT, ACCCCT**. If $|G| = n$ then it is obvious that size of **L** is $(n - 1)^2$.
2. For each element of **L**, count number of times it occurs as substring in **D**. Contribution of Gene **G** in DNA **D** is total of these values. For example if **D = ACTACCTACCCCT** then

AT	0
ACT	1
ACCT	1
ACT	1
ACCT	1
ACCCT	0
ACCT	1
ACCCT	0
ACCCCT	1
Total	6

As this process is very clumsy he wants to automate this process. As he is not a programmer, he needs your help. He will be very grateful if you kindly write him a program which will read the sequence of the DNA and the Gene, and will calculate contribution of the Gene in the DNA.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case contains two lines. The first line contains a string denoting the sequence of DNA, and the second line contains another string denoting the Gene. The length of each string is less than **50000** and consists of only **A, C, T** and **G**.

Output

For each case, print the case number and the contribution, as described above.

Sample Input	Output for Sample Input
3 ACTACCTACCCCT ACCT AAA AAAA AAAA AAA	Case 1: 6 Case 2: 4 Case 3: 8

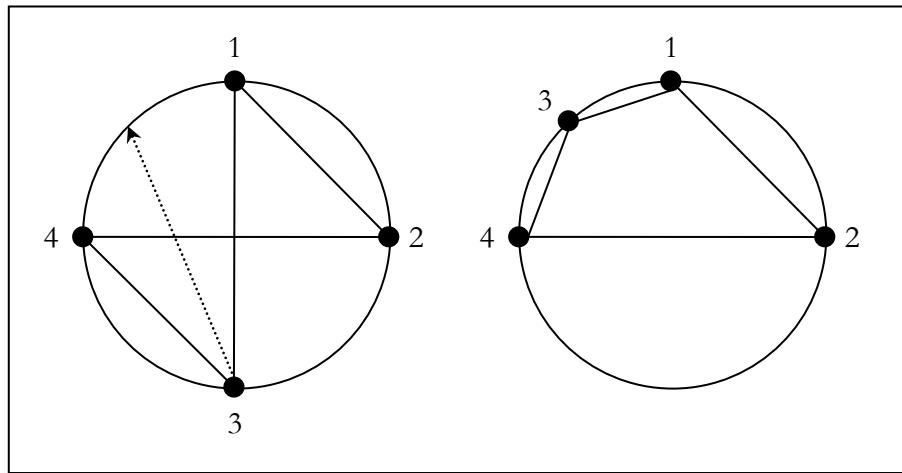
Note

1. Proper prefix (suffix) of a string S is a prefix (suffix) of length smaller than $|S|$. Here $|S|$ denotes length of S .
2. Dataset is huge, use faster I/O methods.

1335 – Planar Graph

A graph is said to be a **planar graph** if it can be drawn on a plane such a way that its edges intersect only at their end points. In other words, it can be drawn in such a way that no edges cross each other.

In this problem, you are given a 2-regular connected graph where the n vertices (numbered from 1 to n) are drawn in the perimeter of a circle in clockwise order. Now the graph may or may not be planar. We want to make it planar. We are allowed to move a vertex to any empty place in the perimeter of the circle. When a vertex is moved, its edges will also move with it, but remain connected with the vertex.



For example, there is a non-planar graph in the left picture, we made it a planar graph (shown in the right picture) by moving vertex-3 between vertex 1 and 4. There are other solutions, but the best we can do is to move one vertex and make it planar. You have to do the similar task, you are given a graph as described, and your task is to make it planar by moving as few vertices as possible. Assume that the circle is large enough such that you can place any number of vertices between two particular vertices.

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a line containing an integer n ($3 \leq n \leq 500$). Each of the next n lines contains two integers. The i^{th} line ($1 \leq i \leq n$) contains the vertices that have an edge with vertex i . You can assume that the given graph follows the restrictions defined above.

Output

For each case, print the case number first. Then if it's impossible to make it planar, print '**impossible**' in the same line. Otherwise, print the minimum possible number of vertices that need to be moved to make the graph planar.

Sample Input	Output for Sample Input
2 4 2 3 4 1 4 1 3 2 6 5 4 5 3 2 6 1 2 1 4 3 6	Case 1: 1 Case 2: 2

Note

1. A graph is said to be **2-regular** if each vertex has exactly two edges.
2. A **connected** graph is an undirected graph where it's possible to move from any vertex to another using the edges.

1336 – Sigma Function

Sigma function is an interesting function in Number Theory. It is denoted by the Greek letter Sigma (σ). This function actually denotes the sum of all divisors of a number. For example $\sigma(24) = 1+2+3+4+6+8+12+24=60$. Sigma of small numbers is easy to find but for large numbers it is very difficult to find in a straight forward way. But mathematicians have discovered a formula to find sigma. If the prime power decomposition of an integer is

$$n = p_1^{e_1} * p_2^{e_2} * \dots * p_k^{e_k}$$

Then we can write,

$$\sigma(n) = \frac{p_1^{e_1+1} - 1}{p_1 - 1} * \frac{p_2^{e_2+1} - 1}{p_2 - 1} * \dots * \frac{p_k^{e_k+1} - 1}{p_k - 1}$$

For some **n** the value of $\sigma(n)$ is odd and for others it is even. Given a value **n**, you will have to find how many integers from **1** to **n** have **even** value of σ .

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 10^{12}$).

Output

For each case, print the case number and the result.

Sample Input	Output for Sample Input
4	Case 1: 1
3	Case 2: 5
10	Case 3: 83
100	Case 4: 947
1000	

1337 – The Crystal Maze

You are in a plane and you are about to be dropped with a parasuit in a crystal maze. As the name suggests, the maze is full of crystals. Your task is to collect as many crystals as possible.

To be more exact, the maze can be modeled as an $\mathbf{M} \times \mathbf{N}$ 2D grid where \mathbf{M} denotes the number of rows and \mathbf{N} denotes the number of columns. There are three types of cells in the grid:

1. A '#' denotes a wall, you may not pass through it.
2. A 'C' denotes a crystal. You may move through the cell.
3. A '!' denotes an empty cell. You may move through the cell.

Now you are given the map of the maze, you want to find where to land such that you can collect maximum number of crystals. So, you are spotting some position \mathbf{x}, \mathbf{y} and you want to find the maximum number of crystals you may get if you land to cell (\mathbf{x}, \mathbf{y}) . And you can only move vertically or horizontally, but you cannot pass through walls, or you cannot get outside the maze.

Input

Input starts with an integer \mathbf{T} (≤ 10), denoting the number of test cases.

Each case starts with a line containing three integers \mathbf{M} , \mathbf{N} and \mathbf{Q} ($2 \leq \mathbf{M}, \mathbf{N} \leq 500, 1 \leq \mathbf{Q} \leq 1000$). Each of the next \mathbf{M} lines contains \mathbf{N} characters denoting the maze. You can assume that the maze follows the above restrictions.

Each of the next \mathbf{Q} lines contains two integers \mathbf{x}_i and \mathbf{y}_i ($1 \leq \mathbf{x}_i \leq \mathbf{M}, 1 \leq \mathbf{y}_i \leq \mathbf{N}$) denoting the cell where you want to land. You can assume that cell $(\mathbf{x}_i, \mathbf{y}_i)$ is empty i.e. the cell contains '!'.

Output

For each case, print the case number in a single line. Then print \mathbf{Q} lines, where each line should contain the maximum number of crystals you may collect if you land on cell $(\mathbf{x}_i, \mathbf{y}_i)$.

Sample Input	Output for Sample Input
1 4 5 2 . . # . . . C # C . # # . . # . . C # C 1 1 4 1	Case 1 : 1 2

Note

Dataset is huge, use faster I/O methods.

1338 - Hidden Secret!

In this problem you are given two names, you have to find whether one name is hidden into another. The restrictions are:

1. You can change some uppercase letters to lower case and vice versa.
2. You can add/remove spaces freely.
3. You can permute the letters.

And if two names match **exactly**, then you can say that one name is hidden into another.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with two lines. Each line contains a name consists of upper/lower case English letters and spaces. You can assume that the length of any name is between **1** and **100** (inclusive).

Output

For each case, print the case number and "**Yes**" if one name is hidden into another. Otherwise print "**No**".

Sample Input	Output for Sample Input
3 Tom Marvolo Riddle I am Lord Voldemort I am not Harry Potter Hi Pretty Roar to man Harry and Voldemort Tom and Jerry and Harry	Case 1: Yes Case 2: Yes Case 3: No

1339 – Strongest Community

In a strange city, houses are built in a straight line one after another. There are several communities in the city. Each community consists of some **consecutive** houses such that every house belongs to **exactly** one community. The houses are numbered from **1** to **n**, and the communities are numbered from **1** to **c**.

Now some inspectors want to find the strongest community considering all houses from **i** to **j**. A community is strongest if maximum houses in the range belong to this community. So, there can be more than one strongest community in the range. So, they want to know the number of houses that belong to the strongest community. That's why they are seeking your help.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

Each case starts with a line containing three integers **n** ($1 \leq n \leq 10^5$), **c** ($1 \leq c \leq n$) and **q** ($1 \leq q \leq 50000$). The next line contains **n** space separated integers (each between **1** and **c**) denoting the communities the houses belong to. You can assume that the input follows the restrictions given above, and there are exactly **c** communities.

Each of the next **q** lines contains two integers **i** and **j** ($1 \leq i \leq j \leq n$) denoting that the inspectors are asking for the result considering houses from **i** to **j** (inclusive).

Output

For each case, print the case number in a single line. Each of the next **q** lines should contain the number of houses that belong to the strongest community considering houses from **i** to **j**. The result should be listed in the same order as they are given in input.

Sample Input	Output for Sample Input
2 10 3 4 1 1 1 3 3 3 3 2 2 2 1 5 1 6 1 7 7 9 3 3 1 3 2 1 1 1	Case 1: 3 3 4 2 Case 2: 1

Note

Dataset is huge, use faster I/O methods.

1340 – Story of Tomisu Ghost

It is now 2150 AD and problem-setters are having a horrified time as the ghost of a problem-setter from the past, Mr. Tomisu, is frequently disturbing them. As always is the case in most common ghost stories, Mr. Tomisu has an unfulfilled dream: he had set 999 problems throughout his whole life but never had the leisure to set the 1000th problem. Being a ghost he cannot set problems now so he randomly asks problem-setters to complete one of his unfinished problems. One problem-setter tried to convince him saying that he should not regret as 999 is nowhere near $1024 (2^{10})$ and he should not worry about power of 10 being an IT ghost. But the ghost slapped him hard after hearing this. So at last one problem setter decides to complete his problem:

"**n!** (factorial **n**) has at least **t** trailing zeroes in **b** based number system. Given the value of **n** and **t**, what is the maximum possible value of **b**?"

Input

Input starts with an integer **T (≤ 4000)**, denoting the number of test cases.

Each case contains two integers **n ($1 < n \leq 10^5$)** and **t ($0 < t \leq 1000$)**. Both **n** and **t** will be given in decimal (base 10).

Output

For each case, print the case number and the maximum possible value of **b**. Since **b** can be very large, so print **b** modulo **10000019**. If such a base cannot be found then print **-1** instead.

Sample Input	Output for Sample Input
4 1000 1000 1000 2 10 8 4 2	Case 1: -1 Case 2: 5227616 Case 3: 2 Case 4: 2

1341 - Aladdin and the Flying Carpet

It's said that Aladdin had to solve seven mysteries before getting the Magical Lamp which summons a powerful Genie. Here we are concerned about the first mystery.

Aladdin was about to enter to a magical cave, led by the evil sorcerer who disguised himself as Aladdin's uncle, found a strange magical flying carpet at the entrance. There were some strange creatures guarding the entrance of the cave. Aladdin could run, but he knew that there was a high chance of getting caught. So, he decided to use the magical flying carpet. The carpet was rectangular shaped, but not square shaped. Aladdin took the carpet and with the help of it he passed the entrance.

Now you are given the area of the carpet and the length of the minimum possible side of the carpet, your task is to find how many types of carpets are possible. For example, the area of the carpet 12, and the minimum possible side of the carpet is 2, then there can be two types of carpets and their sides are: {2, 6} and {3, 4}.

Input

Input starts with an integer **T (≤ 4000)**, denoting the number of test cases.

Each case starts with a line containing two integers: **a b ($1 \leq b \leq a \leq 10^{12}$)** where **a** denotes the area of the carpet and **b** denotes the minimum possible side of the carpet.

Output

For each case, print the case number and the number of possible carpets.

Sample Input	Output for Sample Input
2 10 2 12 2	Case 1: 1 Case 2: 2

1342 – Aladdin and the Magical Sticks

It's said that Aladdin had to solve seven mysteries before getting the Magical Lamp which summons a powerful Genie. Here we are concerned about the second mystery.

Aladdin was going along a magical cave, tricked by the evil sorcerer, searching for the great Magical Lamp. Then he found a strange Lamp, thinking that it might be the lamp the sorcerer had spoken off, rubbed it and a Genie appeared. But alas! He was unlucky as the lamp was owned by an evil Genie. It blindfolded Aladdin and gave him a task to finish. Aladdin finished that task and moved forward.

The task was that, there were **n** sticks, each had a particular weight. Two types of sticks were there, one kind of sticks had distinguishable rough patterns that can be identified by just touching. Other types of sticks were indistinguishable. Each time Aladdin had to pick a stick from all sticks and put it into a magical box. If all the sticks are put into the box **at least once**, Aladdin will be free; otherwise the stick he just put will be put with the other sticks.

So, Aladdin planned that each time he will pick a **new** stick randomly and put it into the magical box. So, once he put a distinguishable stick into the box, he will not put it again though it's mixed with other sticks, since he can remember the roughness of every distinguishable stick. But for indistinguishable sticks he had no option. So, each time he put a stick that looked new to him, but that might not be a new one. And each time the probability of picking a new (to Aladdin of course!) stick is equal. Now your task is to find the **expected** summation of weights of sticks Aladdin had to put into the box before he was free.

For example, let's say there were two sticks, one was distinguishable and the other one was indistinguishable. And the weights were 4, 8 respectively. Let's calculate the expected summation of weights of sticks.

- 1) Aladdin puts stick 1 (weight 4). The probability is $1/2$. As all the sticks are **not** put into the box at least once, so, it will be put with the other stick again. Aladdin will not put stick 1 again, since he can distinguish it from others. So, he puts stick 2 next. And since all the sticks are put in the box at least once, so he is free. Total weight he put is $4 + 8 = 12$. So, the expected summation of weights is $(1/2) * 12 = 6$.
- 2) Let's say he puts stick 2 (weight 8). The probability is $1/2$. Now it will be put back with the other stick. So, Aladdin has two sticks again, where he is not sure which one he had put into the box, as he cannot distinguish stick 2. So, he kept trying stick 1 and 2. And if he puts stick 2 again, he faces the same situation, but if he puts stick 1, he is free as all the sticks are put into the box at least once. Let's say the expected summation of weights of sticks from this situation is **x**. If he puts stick 1 (weight 4, but probability is $1/2$), he is free. If he puts stick 2 (weight 8), he is in the same situation. So, $x = (1/2) * 4 + (1/2) * (x + 8)$. So, $x = 12$. So, the expected summation of weights is $(1/2) * (8 + 12) = 10$.

So, from the both outcomes, the final result is $6 + 10 = 16$.

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 5000$). Each of the next **n** lines contains two integers **a_i b_i** ($1 \leq a_i \leq 5000, 1 \leq b_i \leq 2$) where **a_i** denotes the length of the stick and **b_i** denotes the type. **b_i = 1** indicates that the stick is distinguishable from others, **b_i = 2** means it is not distinguishable.

Output

For each case, print the case number and the expected summation of weights of the sticks Aladdin had to put into the box. Errors less than 10^{-4} will be ignored.

Sample Input	Output for Sample Input
4 2 4 1 8 2 2 5 1 6 1 2 2 2 5 2 3 1 1 2 2 5 2	Case 1: 16 Case 2: 11 Case 3: 10.5000000000 Case 4: 13.8333333333

1343 - Aladdin and the Black Stones

It's said that Aladdin had to solve seven mysteries before getting the Magical Lamp which summons a powerful Genie. Here we are concerned about the third mystery.

In the cave, Aladdin found some black stones arranged in a line. The weights of the stones were not necessarily equal. Though the stones looked gentle, he soon found that he cannot even walk over the stones. Some strange magic were stopping him passing the stones. So, he planned to move the stones. But when he took the first stone, he heard a sound. And it said,

"Remove some stones leaving **even** number of stones, the outer pair should be the heaviest. If the pair is removed, the new outer pair should be the heaviest and if this pair is removed the new outer pair is the heaviest and so on. How many ways you can do? Find it and I will let you through."

Aladdin solved this task and moved forward. Now your task is to do the same, you are given **n** stones W_1, W_2, \dots, W_n , where W_i denotes the weight of the i^{th} stone. You can remove some stones leaving $2m$ ($m > 0$) stones as $W_{p1}, W_{p2}, \dots, W_{p_{2m}}$ such that

$$W_{p1} + W_{p_{2m}} > W_{p2} + W_{p_{2m-1}} > W_{p3} + W_{p_{2m-2}} > \dots > W_{pm} + W_{p_{m+1}}$$

Remember that you cannot change the order of the stones. Now your task is to find the number of ways you can do it. Two orderings are different if **m** is different or there is at least a position **i**, where the stones are different.

Input

Input starts with an integer **T** (≤ 30), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($2 \leq n \leq 500$) and **n** integers denoting the weight of the stones. Weights can be any integer between 1 and 10^9 .

Output

For each case, print the case number and the number of possible orderings modulo 2^{32} .

Sample Input	Output for Sample Input
4 3 1 1 1 4 1 4 7 20 4 10 18 2 9 7 2 4 1 4 5 9 13	Case 1: 3 Case 2: 7 Case 3: 6 Case 4: 61

Note

1. For the first case, the valid orderings are $\{1_{(\text{first stone})}, 1_{(\text{second stone})}\}$, $\{1_{(\text{first stone})}, 1_{(\text{third stone})}\}$ and $\{1_{(\text{second stone})}, 1_{(\text{third stone})}\}$.
2. For the second case, the valid orderings are, $\{1, 4\}$, $\{1, 7\}$, $\{1, 20\}$, $\{4, 7\}$, $\{4, 20\}$, $\{7, 20\}$, $\{1, 4, 7, 20\}$.

1344 - Aladdin and the Game of Bracelets

It's said that Aladdin had to solve seven mysteries before getting the Magical Lamp which summons a powerful Genie. Here we are concerned about the fourth mystery.

In the cave, Aladdin was moving forward after passing the pathway of magical stones. He found a door and he was about to open the door, but suddenly a Genie appeared and he addressed himself as the guardian of the door. He challenged Aladdin to play a game and promised that he would let Aladdin go through the door, if Aladdin can defeat the Genie. Aladdin defeated the Genie and continued his journey. However, let's concentrate on the game.

The game was called 'Game of Bracelets'. A bracelet is a linear chain of some pearls of various weights. The rules of the game are:

- 1) There are **n** bracelets.
- 2) Players alternate turns.
- 3) In each turn, a player has to choose a pearl from any bracelet. Then all the pearls from that bracelet, that were **not lighter** than the pearl, will be removed. It may create some smaller bracelets or the bracelet will be removed if no pearl is left in the bracelet.
- 4) The player, who cannot take a pearl in his turn, loses.

For example, two bracelets are: 5-1-7-2-4-5-3 and 2-1-5-3, here the integers denote the weights of the pearls. Suppose a player has chosen the first pearl (weight 5) from the first bracelet. Then all the pearls that are not lighter than 5, will be removed (from first bracelet). So, 5-1-7-2-4-5-3, the red ones will be removed and thus from this bracelet, three new bracelets will be formed, 1, 2-4 and 3. So, in the next turn the other player will have four bracelets: 1, 2-4, 3 and 2-1-5-3. Now if a player chooses the only pearl (weight 3) from the third bracelet, then the bracelet will be removed.

Now you are given the information of the bracelets. Assume that Aladdin plays first, and Aladdin and the Genie both play optimally. Your task is to find the winner.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 50$)**. Each of the next **n** lines contains an integer **K_i ($1 \leq K_i \leq 50$)** followed by **K_i** integers, denoting the weights of the pearls of the **ith** ($1 \leq i \leq n$) bracelet. Each weight will lie in the range **[1, 10⁵]**.

Output

For each case, print the case number and '**Aladdin**' if Aladdin wins. Otherwise print '**Genie**'. If Aladdin wins, then print an additional line, denoting the weight of pearls, one of which should be chosen in the **first** move by Aladdin such that he can win. Each pearl should be denoted by a pair of integers: the first integer is the bracelet number and the second integer is the weight. Sort the pearls first by the bracelet number then by the weight. And same weight in a bracelet should be reported once. Check the samples for exact formatting.

Sample Input	Output for Sample Input
<pre> 4 2 7 5 1 7 2 4 5 3 4 2 1 5 4 2 2 5 2 2 5 2 1 5 5 2 5 2 5 3 5 5 2 5 2 5 5 7 2 7 3 2 4 5 1 5 4 </pre>	<pre> Case 1: Aladdin (2 5) Case 2: Genie Case 3: Aladdin (1 2) (1 5) Case 4: Aladdin (2 7) (3 1) (3 5) </pre>

1345 - Aladdin and the Happy Garden

It's said that Aladdin had to solve seven mysteries before getting the Magical Lamp which summons a powerful Genie. Here we are concerned about the fifth mystery.

After defeating the Genie in 'Game of Bracelets', Aladdin moved forward and found a garden. There were **n** small Genies in the garden and they were all sad. Aladdin asked them about the lamp. But none of them was interested. So, Aladdin had to make them happy. Soon he noticed that height of every Genie is distinct but they were standing in a line randomly. Aladdin found a book there, and after reading the book, he discovered that the only way to make them happy was to make a line with the Genies such that **k consecutive** genies in the line could be found whose heights are in increasing order. But the book also mentioned that if **k+1 consecutive** Genies found in the line whose heights are in increasing order then they would be sad again. So, Aladdin did make them happy, and they showed the path to Aladdin and he moved forward.

Now you are given **n** and **k**, your task is to find in how many ways Aladdin could make them happy.

Input

Input starts with an integer **T (≤ 1275)**, denoting the number of test cases.

Each case starts with a line containing two integers: **n** and **k ($1 \leq k \leq n \leq 50$)**.

Output

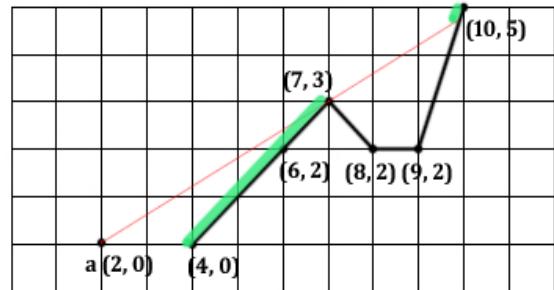
For each case, print the case number and the number of ways Aladdin could make them happy. As the result can be big; print the result modulo **1000 000 007**.

Sample Input	Output for Sample Input
5 2 2 3 2 4 3 5 4 5 3	Case 1: 1 Case 2: 4 Case 3: 6 Case 4: 8 Case 5: 41

1346 – Aladdin and the Rocky Mountains

It's said that Aladdin had to solve seven mysteries before getting the Magical Lamp which summons a powerful Genie. Here we are concerned about the sixth mystery.

After the Happy Garden, Aladdin reached a place where he found a huge area of Rocky Mountains. He could barely see other than the rocks. His target was to reach the highest pick, such that all parts could be observed by him. He was searching for the layer of 3-Eyed Monsters where the lamp was located. So, he started climbing and finally managed to find the layer.



However, here we are concerned about a similar problem. Assume that the mountains are described by 2D co-ordinates and have negligible thickness. Aladdin is standing in a position which is $(a_x, 0)$; your target is to find the area that can be seen by him from this position. Aladdin cannot see a point if there is any point between him and the object point. For example, in the picture, Aladdin's coordinate is $(2, 0)$ and the points $(4, 0)$, $(6, 2)$, $(7, 3)$, $(8, 2)$, $(9, 2)$ and $(10, 5)$ form the mountain. Y coordinate 0 means it's ground. He can see the green sides (as in picture). You have to find this area.

Input

Input starts with an integer T (≤ 30), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers n ($2 \leq n \leq 20000$) and a_x , n denotes the number of picks in the mountain, and a_x denotes the x co-ordinate of Aladdin. Each of the next n lines contains two integers: x_i and y_i forming the mountain. Assume that $x_i > a_x$, $y_1 = 0$ and $x_i > x_{i-1}$ for $2 \leq i \leq n$. Values for the co-ordinates will lie in the range $[0, 50000]$.

Output

For each case, print the case number and the total area of the Mountain sides (above ground) that can be observed by Aladdin. Errors less than 10^{-3} will be ignored.

Sample Input	Output for Sample Input
1 6 2 4 0 6 2 7 3 8 2 9 2 10 5	Case 1: 4.5061638255

Note

Dataset is huge, use faster I/O methods.

1347 – Aladdin and the Magical Lamp

It's said that Aladdin had to solve seven mysteries before getting the Magical Lamp which summons a powerful Genie. Here we are concerned about the seventh and final mystery.

Finally, Aladdin reached the layer of three eyed monsters, who were too dangerous. He was moving quietly without being seen. However, he was not lucky enough and some monsters found him. So, Aladdin had no option but to fight. But the monsters were too strong that Aladdin couldn't defeat them. But soon he realized that he could only beat them using some other technique. He discovered that, each eye of the three eyed monster had a rough pattern. He found the longest common sub-pattern in three eyes, and attacked the common patterns in all eyes. Surprisingly he found that the monster got weaker and weaker. And it was dead when Aladdin removed all the patterns.

And finally, Aladdin managed to kill all the monsters, and found the Lamp in a lotus. When he rubbed it, a Genie was summoned, saying, "Master, I am here, I will fulfill your three wishes". And the quest for the great magical lamp was at end, and became a history!

Here, you are given the same problem that was solved by Aladdin; that is you are given three patterns; you have to find the longest common sub-pattern in **all** the patterns. This sub-pattern should contain some consecutive symbols from the original patterns. For example, let the patterns be "aladdin", "adding", "dinner", then the longest sub-pattern in all the pattern is "din".

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. Each of the next three lines contains a pattern. A pattern is a **non-empty** string consisting of lower-case English letters whose length is no more than **5000**.

Output

For each case, print the case number and the size of the longest common sub-pattern that occurs in all the patterns.

Sample Input	Output for Sample Input
2 aladdin adding dinner math magic matters	Case 1: 3 Case 2: 2

Note

This problem was used in the Contest - [Aladdin's Journey](#), but the max length of a string was 20.

1348 – Aladdin and the Return Journey

Finally the Great Magical Lamp was in Aladdin's hand. Now he wanted to return home. But he didn't want to take any help from the Genie because he thought that it might be another adventure for him. All he remembered was the paths he had taken to reach there. But since he took the lamp, all the genies in the cave became angry and they were planning to attack. As Aladdin was not afraid, he wondered how many genies were there. He summoned the Genie from the lamp and asked this.

Now you are given a similar problem. For simplicity assume that, you are given a tree (a connected graph with no cycles) with n nodes, nodes represent places, edges represent roads. In each node, initially there are an arbitrary number of genies. But the numbers of genies change in time. So, you are given a tree, the number of genies in each node and several queries of two types. They are:

- 1) **0 i j**, it means that you have to find the total number of genies in the **nodes** that occur in path from node **i** to **j** ($0 \leq i, j < n$).
- 2) **1 i v**, it means that number of genies in node **i** is changed to **v** ($0 \leq i < n, 0 \leq v \leq 1000$).

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with a blank line. Next line contains an integer **n** ($2 \leq n \leq 30000$). The next line contains **n** space separated integers between **0** and **1000**, denoting the number of genies in the nodes respectively. Then there are **n-1** lines each containing two integers: **u v** ($0 \leq u, v < n, u \neq v$) meaning that there is an edge from node **u** and **v**. Assume that the edges form a valid tree. Next line contains an integer **q** ($1 \leq q \leq 10^5$) followed by **q** lines each containing a query as described above.

Output

For each case, print the case number in a single line. Then for each query **0 i j**, print the total number of genies in the nodes that occur in path **i** to **j**.

Sample Input	Output for Sample Input
1 4 10 20 30 40 0 1 1 2 1 3 3 0 2 3 1 1 100 0 2 3	Case 1: 90 170

Note

Dataset is huge, use faster I/O methods.

1349 – Aladdin and the Optimal Invitation

Finally Aladdin reached home, with the great magical lamp. He was happier than ever. As he was a nice boy, he wanted to share the happiness with all people in the town. So, he wanted to invite all people in town in some place such that they can meet there easily. As Aladdin became really wealthy, so, number of people was not an issue. Here you are given a similar problem.

Assume that the town can be modeled as an $m \times n$ 2D grid. People live in the cells. Aladdin wants to select a cell such that all people can gather here with optimal **overall cost**. Here, cost for a person is the distance he has to travel to reach the selected cell. If a person lives in cell (x, y) and he wants to go to cell (p, q) , then the cost is $|x-p| + |y-q|$. So, distance between $(5, 2)$ and $(1, 3)$ is $|5-1| + |2-3|$ which is 5. And the **overall cost** is the **summation** of costs for all people.

So, you are given the information of the town and the people, your task to report a cell which should be selected by Aladdin as the gathering point and the **overall cost** should be as low as possible.

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers: m , n and q ($1 \leq m, n, q \leq 50000$), m and n denote the number of rows and columns of the grid respectively. Each of the next q lines contains three integers $u v w$ ($1 \leq u \leq m$, $1 \leq v \leq n$, $1 \leq w \leq 10000$), meaning that there are w persons who live in cell (u, v) . You can assume that there are no people in the cells which are not listed. You can also assume that each of the q lines contains a distinct cell.

Output

For each case, print the case number and the row and column position of the cell where the people should be invited. There can be multiple solutions, any valid one will do.

Sample Input	Output for Sample Input
2 5 1 1 2 1 10 5 5 4 1 1 1 2 2 1 4 4 1 5 5 1	Case 1: 2 1 Case 2: 3 3

Note

1. This is a special judge problem; wrong output format may cause 'Wrong Answer'.
2. Dataset is huge, use faster I/O methods.

1350 – Aladdin and the Grand Feast

As Aladdin invited all people in the town to share his happiness and joy, all people came to meet him. And he arranged a **Grand Feast** for them. There were many delicious food items, fruits, drinks, sweets and a lot more.

As like a regular function, each person had a particular arrival time (a_i) and departure time (d_i). It means, he can start eating in time a_i , and cannot eat after time $d_i - 1$. And of course not all of them wanted to have the same amount of food. So, a person wanted to eat **exactly** f_i unit of food and of course within his arrival and departure time. Note that a person can have one unit of food in one time unit.

But there were only t tables, and each table had c chairs. So, in a single time unit, at most $t * c$ people can eat together. But in the next time unit, a new group of people can replace them (takes negligible amount of time). So, Aladdin made a schedule (when to eat and where to sit) for the persons such that all became happy.

Now your task is to do the same. You are given all the necessary information, your task is to find a schedule for the persons such that everyone becomes happy. A person is happy if he gets exactly f_i unit of food in his time interval. Assume that the feast will last long until all the persons are gone.

Input

Input starts with an integer T (≤ 30), denoting the number of test cases.

Each case starts with a blank line. Next line contains four integers: $n \ t \ c \ e$ ($1 \leq n \leq 50, 1 \leq t, c \leq 5, 2 \leq e \leq 10^4$), where n denotes the number of invited people and e denotes the end time of the feast. Each of the next n lines contains three integers: $a_i \ d_i \ f_i$ ($1 \leq a_i < d_i \leq e, 1 \leq f_i \leq d_i - a_i$).

Output

For each case, print the case number and '**Yes**' if it's possible to make everyone happy. Print '**No**' otherwise.

If the result is yes, then print additional $e-1$ lines. The i^{th} line should contain the persons sitting in the tables in i^{th} time in the following format:

- 1) First 26 persons should be identified by the lower case English letters; the next ones should be identified by capital letters (in same order). So, person 6 is '**f**', and person 28 is '**B**'.
- 2) Each table should be denoted by c characters, where a '.' denotes the chair in that table is empty; otherwise it should contain the person who is eating here. Print all the tables in a single line. Two tables should be separated by a '|' character. See the samples for more details.

There can be multiple solutions, any valid one will do.

Sample Input	Output for Sample Input
<pre> 3 1 1 2 3 1 3 2 7 2 3 5 1 4 3 1 5 4 1 5 4 1 2 1 2 4 2 2 5 2 3 5 1 4 1 3 3 1 3 2 1 3 1 1 3 2 1 3 2 </pre>	<pre> Case 1: Yes .a .a Case 2: Yes ..a bcd .ab cef ..a bce ..b cfg Case 3: No </pre>

Note

- 1) Dataset is huge, use faster I/O methods.
- 2) This is a special judge problem; wrong output format may cause 'Wrong Answer'.

1351 – Ordered Flips

You are given two non-empty strings **X** and **Y** of same length **n**. Your task is to make them identical. But the problem is that the only operation you can do is flipping, and it can only be applied to **X**.

For a flip, two positions of the string **X** are chosen, let the positions be **i** and **j** ($0 \leq i < j < n$) and if you apply **flip (i, j)** all characters between **i** and **j** (inclusive) are reversed. For example, let **X** be "**abcdefg**", then if you apply flip (2, 5) to **X** then **X** will be "**abfedcg**". But if you want to apply flips more than once, you have to use ordered flips. If **flip (i₂, j₂)** is applied immediately after **flip (i₁, j₁)**, then it will be said "**Ordered Flips**" if and only if $i_1 \leq i_2$ and $j_2 \leq j_1$.

So, now your task is to find the minimum number of ordered flips to change **X** to **Y**.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case contains two lines, each containing a non empty string of length **n** ($1 \leq n \leq 60$). The strings contain lowercase English letters only. First line contains **X** and second line contains **Y**.

Output

For each case, print the case number and the minimum number of ordered flips needed to change **X** to **Y**. If it's impossible to do, then print "**impossible**". Check the samples for details.

Sample Input	Output for Sample Input
4 abcd dcba abca aabc zzzaaazzzaaa aazzaazzzaazz aab bab	Case 1: 1 Case 2: 2 Case 3: 4 Case 4: impossible

1352 - Strange Summation

Mr Foolizm is learning binary numbers. He has a software that converts any decimal integer to binary and vice versa. So, if he is asked to add some decimal numbers, he first converts them to binary. Then he adds them in binary. But he doesn't understand the carry principle yet and he even doesn't know the idea of positioning the numbers. Instead of LSD (least significant digit), he starts summing from MSD (most significant digit). So, if he is asked to add four integers from 1 to 4, he does the following:

1. He first takes 1 and 2 and convert them to binary using his software, and then he adds them like the following:

```
1  
10  
----  
00
```

2. Then he takes 3, converts it to binary and add it with the previous result

```
00  
11  
----  
11
```

3. He then takes 4, converts it to binary and add it with the previous result

```
11  
100  
----  
010
```

4. And finally he converts the result back to decimal, so, the result is 2.

Now you are given two integers **p** and **q**, your task is to find the result if Mr Foolizm adds all integers from **p** to **q** (inclusive) in his procedure.

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case starts with a line containing two integers: **p** and **q** ($0 < p < q < 2^{63}$).

Output

For each case, print the case number and the result.

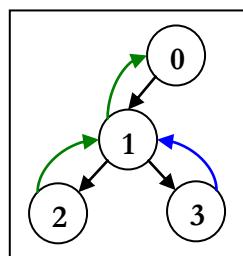
Sample Input	Output for Sample Input
4 1 4 2 5 1 2147483647 7 12	Case 1: 2 Case 2: 3 Case 3: 1610612736 Case 4: 2

1353 – Paths in a Tree

You are given a tree (a connected graph with no cycles), and the edges of the tree which are for some reason directed; your task is to add **minimum** number of special paths in the tree such that it's possible to go from any node to another. The rules for the special paths are noted below:

1. A special path consists of some continuous edges (from the tree) and nodes.
2. In a special path, the edges should be in opposite directions as they are in the tree.
3. A node or an edge can be visited at most once in a special path.
4. Multiple special paths may have common nodes or edges.

For example, in the picture below, a tree is drawn, the black arrows represent the edges and their directions, circles represent nodes. Then we need two special paths. One path is **2-1-0** (green arrow), another is **3-1** (blue arrow). Instead of the path **3-1** we can add **3-1-0**. You cannot add a path like **1-3** or **0-1-2** because of rule 2. You cannot add **0-2** or **2-3-0** because of rule 1.



Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing an integer **N ($2 \leq N \leq 20000$)**, where **N** denotes the number of nodes. The nodes are numbered from **0** to **N-1**. Each of the next **N-1** lines contains two integers **u v ($0 \leq u, v < N, u \neq v$)** meaning that there is an edge from **u** to **v**.

Output

For each case, print the case number and the minimum number of special paths required such that it's possible to go from any node to another.

Sample Input	Output for Sample Input
2 4 0 1 1 2 1 3 5 0 1 1 2 1 3 0 4	Case 1: 2 Case 2: 3

1354 - IP Checking

An IP address is a 32 bit address formatted in the following way

a.b.c.d

where **a, b, c, d** are integers each ranging from **0 to 255**. Now you are given two IP addresses, first one in decimal form and second one in binary form, your task is to find if they are same or not.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with two lines. First line contains an IP address in decimal form, and second line contains an IP address in binary form. In binary form, each of the four parts contains 8 digits. Assume that the given addresses are valid.

Output

For each case, print the case number and "**Yes**" if they are same, otherwise print "**No**".

Sample Input	Output for Sample Input
2 192.168.0.100 11000000.10101000.00000000.11001000 65.254.63.122 01000001.1111110.00111111.01111010	Case 1: No Case 2: Yes

1355 – Game of CS

Jolly and Emily are two bees studying in Computer Science. Unlike other bees they are fond of playing two-player games. They used to play Tic-tac-toe, Chess etc. But now since they are in CS they invented a new game that definitely requires some knowledge of computer science.

Initially they draw a random rooted tree (a connected graph with no cycles) in a paper which consists of n nodes, where the nodes are numbered from **0** to **n-1** and **0** is the root, and the edges are weighted. Initially all the edges are unmarked. And an edge weigh **w**, has **w** identical units.

1. Jolly has a green marker and Emily has a red marker. Emily starts the game first and they alternate turns.
2. In each turn, a player can color **one unit** of an edge of the tree if that edge has some (at least one) uncolored units and the edge can be traversed from the root using only free edges. An edge is said to be free if the edge is not fully colored (may be uncolored or partially colored).
3. If it's Emily's turn, she finds such an edge and colors one unit of it using the red marker.
4. If it's Jolly's turn, he finds such an edge and colors one unit of it with the green marker.
5. The player, who can't find any edges to color, loses the game.

For example, Fig 1 shows the initial tree they have drawn. The tree contains four nodes and the weights of the edge **(0, 1)**, **(1, 2)** and **(0, 3)** are 1, 1 and 2 respectively. Emily starts the game. She can color any edge she wants; she colors one unit of edge **(0 1)** with her red marker (Fig 2). Since the weight of edge **(0 1)** is 1 so, this edge is fully colored.

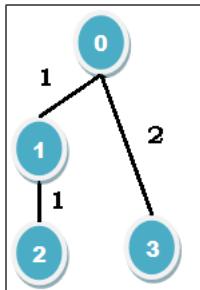


Fig 1

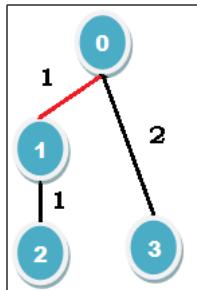


Fig 2

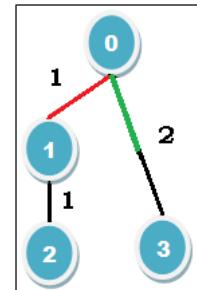


Fig 3

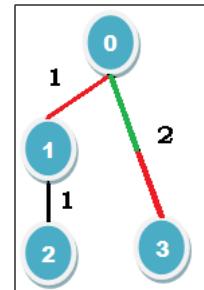


Fig 4

Now it's Jolly's turn. He can only color one unit of edge **(0 3)**. He can't color edge **(1 2)** since if he wants to traverse it from the root **(0)**, he needs to use **(0, 1)** which is fully colored already. So, he colors one unit of edge **(0 3)** with his green marker (Fig 3). And now Emily has only one option and she colors the other unit of **(0 3)** with the red marker (Fig 4). So, both units of edge **(0 3)** are colored. Now it's Jolly's turn but he has no move left. Thus Emily wins. But if Emily would have colored edge **(1 2)** instead of edge **(0 1)**, then Jolly would win. So, for this tree Emily will surely win if both of them play optimally.

Input

Input starts with an integer **T** (≤ 500), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($2 \leq n \leq 1000$). Each of the next **n-1** lines contains two integers **u v w** ($0 \leq u, v < n, u \neq v, 1 \leq w \leq 10^9$) denoting that there is an edge between **u** and **v** and their weight is **w**. You can assume that the given tree is valid.

Output

For each case, print the case number and the name of the winner. See the samples for details.

Sample Input	Output for Sample Input
4 4 0 1 1 1 2 1 0 3 2 5 0 1 1 1 2 2 0 3 3 0 4 7 3 0 1 1 0 2 1 4 0 1 1 1 2 1 1 3 1	Case 1: Emily Case 2: Emily Case 3: Jolly Case 4: Emily

Note

Dataset is huge, use faster I/O methods.

1356 – Prime Independence

A set of integers is called prime independent if none of its member is a prime multiple of another member. An integer **a** is said to be a **prime multiple** of **b** if,

a = b x k (where **k** is a prime [1])

So, **6** is a prime multiple of **2**, but **8** is not. And for example, **{2, 8, 17}** is prime independent but **{2, 8, 16}** or **{3, 6}** are not.

Now, given a set of distinct positive integers, calculate the largest prime independent subset.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with an integer **N ($1 \leq N \leq 40000$)** denoting the size of the set. Next line contains **N** integers separated by a single space. Each of these **N** integers are distinct and between **1** and **500000** inclusive.

Output

For each case, print the case number and the size of the largest prime independent subset.

Sample Input	Output for Sample Input
3 5 2 4 8 16 32 5 2 3 4 6 9 3 1 2 3	Case 1: 3 Case 2: 3 Case 3: 2

Note

1. An integer is said to be a prime if it's divisible by exactly two distinct integers. First few prime numbers are **2, 3, 5, 7, 11, 13, ...**
2. Dataset is huge, use faster I/O methods.

1357 – Corrupted Friendship

Many of us know a person named Mr. Haba Kom Khan. He is very corrupted and lazy. He maintains links with a lot of (corrupted) persons. In order to corrupt more lazy persons, he invites his friends in a recursive fashion in a meeting. The idea is:

If **X** has some cards, he keeps one card and he invites one of his friends **Y** who has no invitation cards yet and **X** gives all the remaining cards to **Y**. Then we say that **Y** is invited by **X**. **Y** then has the responsibility to distribute as many cards as he can; **Y** keeps one card and continues the same procedure as **X**. When **Y** cannot find any more friends to give the invitation cards, he gives the remaining cards back to **X**. **X** then checks for his friends who have no cards yet. If any such friend is found, **X** continues the same procedure. Otherwise, if **X** was invited by **Z** then **X** gives the remaining cards back to **Z**. **X** keeps the cards if there is no such person.

Initially Mr. Haba has **N** invitations cards, and he starts the invitation process as stated. There are **N** persons, and they are numbered from **1** to **N**. Mr. Haba is the person numbered **1**. And the strange fact is that, after the invitation process, each person gets exactly **1** card.

Given all the information of persons being invited by others, Mr. Haba wants you to find the total number of invitations that was made. He also asks you to find the number of different pairs of persons who are certainly not friends. Help Mr. Haba Kom Khan to succeed in his corrupted life!

Input

Input starts with an integer **T** (≤ 30), denoting the number of test cases.

Each case starts with an integer **N** ($1 \leq N \leq 10^5$). Each of the next **N-1** lines will contain two integers, **X** and **Y** ($1 \leq X, Y \leq N, X \neq Y$) denoting that person **Y** received his invitation card from person **X**.

Output

For each case, print the case number, total number of invitations made, and the number of different pairs of persons who are surely not friends. See samples for detailed formatting.

Sample Input	Output for Sample Input
2 2 1 2 3 1 2 1 3	Case 1: 1 0 Case 2: 2 1

Note

Dataset is huge, use faster I/O methods.

1358 – Fukushima Nuclear Blast

One of the most disastrous nuclear accidents in history has taken place at Japan's Fukushima Daiichi nuclear plant. Reactor buildings have been rocked by explosions, caused after damage was sustained from a massive earthquake and tsunami on 11th march and thus releasing dangerous radiation of unspecified proportions into the air.

The radiation is spreading and it's not a threat only to Japan but also to other countries. If the level of radiation reaches a high level, it will surely cause harms to human health as well as the environment.

You, one of the great programmers in the city, have planned to find the zones that are vulnerable to radiation. If you can measure the level of radiation in a certain city, further actions can be taken to prevent the people from radiation related health consequences. So, at first you want to find the time in which a certain percentage of an area is under radiation threat.



So, at first you modeled the map of a city in **2D** space as a **simple polygon** [1]. You denoted the origin of the explosion as a single point. From this origin, the radiation spreads circularly. You plotted the map such that in each unit of time the radius of the radiation grows one unit. Now, you want to find the time when **P%** of the area of a city is under threat of radiation.

Input

Input starts with an integer **T** (≤ 70), denoting the number of test cases.

Each test case starts with a blank line. The next line contains an integer **n** ($3 \leq n \leq 5000$) denoting the number of vertices of the polygon. Each of the next **n** lines will contain two integers **x_i, y_i** denoting the co-ordinate of a vertex of the polygon. The vertices will be given in anticlockwise order. The next line contains 3 integers **r_x, r_y** and **P** ($0 < P \leq 100$), where (r_x, r_y) denotes the co-ordinate of the origin of explosion and **P** denotes the percentage. All values for the co-ordinates are between **-200** to **200** (inclusive).

Output

For each case, print the case number and the time when exactly **P%** of the total area is under threat of radiation. Round the time to nearest integer.

Sample Input	Output for Sample Input
<pre> 2 3 -5 0 5 0 0 5 0 0 100 4 0 0 5 0 3 1 5 6 0 0 17 </pre>	<pre> Case 1: 5 Case 2: 2 </pre>

Note

1. In geometry, a **simple polygon** is a closed polygonal chain of line segments in the plane which do not have points in common other than the common vertices of pairs of consecutive segments.
2. The judge data is prepared such that the result (before rounding) will not be between **x.3 to x.7**. For example, the result can be like 5.8, 24.3 or 81.791. But the result will not be like 24.5, 78.4, etc.

1359 – Sabotaging Contest

Programming contests are now quite frequent in Bangladesh. Just 5 years ago, you could participate in only one or two contests per year, but now, there is one in almost every month. Maybe, someday, you will see contests in every week, or maybe every day.

Due to the schedule of the contests, the contestants are really conscious about optimizing the time spent in travelling. For example, if the contest starts at 9:00AM, they plan to reach contest venue, just before that time, so that, no time is wasted. They will be using various forms of public transportsations like bus, train, taxi, rickshaw to reach the contest spot. Bus and train may be used to transport between cities, taxi and rickshaws may be used to move within cities. You will be given all the routes one can use to move between cities. Since, contestants are optimizing their time; they will always take the fastest route possible. Rickshaws and taxis always use the fastest route, so you don't have to be concerned about them.



As the programming contests have become popular, it also caught interest of the mafias as well. They are planning to sabotage the **next** contest. For that, they will call strike in any city, and thus, no vehicle will enter or leave the city. You can't even use rickshaws and taxi to move in the city during the strike. So, it won't be possible for any team to travel in the city, and may have to use a longer route. If any team can't use their **shortest route**, they will surely be late, and thus, will not be able to attend the contest.

Mafia lords have a list of teams. They will call strike in a city (exactly one); so that these teams will not be able to attend the contest. There may be more than one option. However, you have somehow managed to know about this evil plan. Now, given the list of teams, you have to find the teams who will be affected by it.

Input

Input starts with an integer **T (≤ 7)**, denoting the number of test cases.

Each test case starts a blank line. Next line contains two integers **N, M ($1 \leq N \leq 10^5, 0 \leq M \leq 5*10^5$)** the number of cities and the number of direct routes between cities respectively. Each city is denoted by an integer between **0** and **N-1**. The contest is going to be held in city **0**. Each of the next **M** lines contains three integers, **u, v, t ($0 \leq u, v < N, u \neq v, 1 \leq t \leq 1000$)**, a route between city **u** and **v** that takes time **t**. There will be at most one direct route between two cities.

This is followed by an integer **Q ($1 \leq Q \leq 1000$)**, the number of queries. Each of the next **Q** lines describes a query. Each query starts with an integer **K ($1 \leq K \leq 100$)**, the number of teams in the list, followed by **K** distinct integers: **n₁ n₂, ..., n_k ($0 \leq n_i < N$)** denoting the cities, all teams from these cities should not attend the contest.

Output

For each test case, output the case number in a single line. Then for each query, print a single line, containing a **0** if sabotaging is not necessary. Otherwise print two integers **x y**, where **x** denotes the number of cities where they can call strikes and **y** denotes the number of teams that will surely miss the contest if any of the city is chosen for sabotage.

Sample Input	Output for Sample Input
3 9 10 0 1 10 1 3 5 3 5 10 5 4 5 4 1 20 4 2 10 2 0 40 6 7 7 8 6 8 6 4 30 3 2 8 7 3 5 2 6 2 5 4 6 4 0 1 10 1 2 10 2 3 10 3 4 10 1 2 4 2 3 1 0 1 10 1 1 2	Case 1: 4 3 1 9 2 7 Case 2: 3 3 Case 3: 0

Notes

1. Dataset is huge, use faster I/O methods.
2. For the 1st query in 1st case, as they have to forbid teams from city 7 and 8; they can choose one of the cities from {0, 1, 4, 6}, and if any of them is chosen, teams from cities {6, 7, 8} are sure that they will not be able to attend the contest.

1360 - Skyscraper

The Build n' Profit construction company is about to build its tallest building. It will be huge, the tallest building in the world by a wide margin. It will house hundreds of thousands of people and have rocket-powered elevators to the upper floors. They even plan for a shuttle docking station instead of a helipad on its roof. But any building of that size is extremely costly to build and maintain, and even after selling and renting out all the floor-space it will be very difficult to meet the costs. Luckily, they have come up with a great solution. They will place advertisements on the outer walls of the building for a hefty charge. This will help offset some of the costs and bring in a profit.

However, feedbacks from prospective buyers of this advertisement space have brought up a new problem. Each customer wants a specific sized advertisement placed at a specific height, and they will pay a certain amount of money for it. Each advertisement order specifies its position (i.e. the lowest floor of the advertisement) and its size (i.e. the number of floors it covers, including the starting floor). Each advertisement spans the whole face of the building, so no two advertisements can occupy the same floor and no floors can be partially covered. Each order also includes the amount to be earned if that advertisement is placed on the building. Of course, no money is earned if only part of an advertisement is placed, or it is placed in any other position.

Since many of the advertisements want some of the same floors as others, it is often impossible to choose all of them. Can you help choosing which of the orders to accept so that the above constraints are fulfilled and the amount of profit is maximized?

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with an integer **N ($1 \leq N \leq 30000$)** denoting the number of advertisement orders. Each of the next **N** lines represents an advertisement by three integers **A ($0 \leq A \leq 10^5$)**, **B ($1 \leq B \leq 10^5$)** and **C ($1 \leq C \leq 1000$)** denoting the lowest floor, the number of floors the advertisement covers (including the lowest floor) and the amount of money earned for placing it, respectively.

Output

For each case, print the case number and the maximum profit they can achieve.

Sample Input	Output for Sample Input
1 3 1 5 1 2 10 3 7 12 1	Case 1: 3

Note

Dataset is huge, use faster I/O methods.

1361 - Component Placement

In circuit design, component placement is an important step in the design automation. Inferior placements may affect the performance and manufacturing cost.

You are given a PCB (Printed Circuit Board). It's a large green board. You can place components on either side of the PCB. However, cost of placing a component on both sides are not the same.

You are given n components. For each component c_i , cost of placing it on the top layer is X_i and on the bottom layer is Y_i . These components may interact with each other. If both the components are on the same side of the PCB, the interconnection cost is negligible. But, if they are on different sides, their interconnection is costly. For each such interconnection j , the cost will be Z_j .

Finally, some design issues restricts some components to be on the top side or bottom side. Now, find the minimum cost to place the components.

Input

Input starts with an integer T (≤ 35), denoting the number of test cases.

Each case starts with a line containing two integers N and M ($1 \leq N \leq 200$, $0 \leq M \leq N*(N-1)/2$), where N denotes the number of components, M denotes the number of interconnections.

Then there will be two lines, first line contains N integers, where i^{th} of it describes the cost of placing the component on the top layer. The next line contains N integers denoting the costs for placing the components on the bottom layer respectively. These integers will lie in the range $[1, 10^7]$.

The next line contains N integers ranging in $[-1, 1]$ where -1 denotes the i^{th} component must be placed on bottom, $+1$ means it must be placed on top and 0 means it can be placed on either side.

Then there will be M lines, each containing three integers p q r ($1 \leq p, q \leq N$, $p \neq q$, $1 \leq r \leq 10^7$), denoting that, p^{th} and q^{th} component has to be interconnected and if they are on different layers, the cost of interconnection will be r . There will be at most one interconnection between any pair of components.

Output

For each case, print the case number and the minimum cost to place the components.

Sample Input	Output for Sample Input
5 4 0 5 6 7 8 8 7 6 5 0 0 0 0 4 2 5 6 7 8 8 7 6 5 0 0 0 0 1 3 10 2 4 10 4 3 5 6 7 8 8 7 6 5 0 0 0 0 1 3 10 2 4 10 2 3 1 4 3 5 6 7 8 30 31 32 33 0 0 0 0 1 3 10 2 4 10 2 3 1 4 3 5 6 7 8 8 7 6 5 -1 0 0 1 1 2 10 3 4 10 2 3 1	Case 1: 22 Case 2: 24 Case 3: 25 Case 4: 26 Case 5: 31

Note

Dataset is huge, use faster I/O methods.

1362 – Electricity Connection

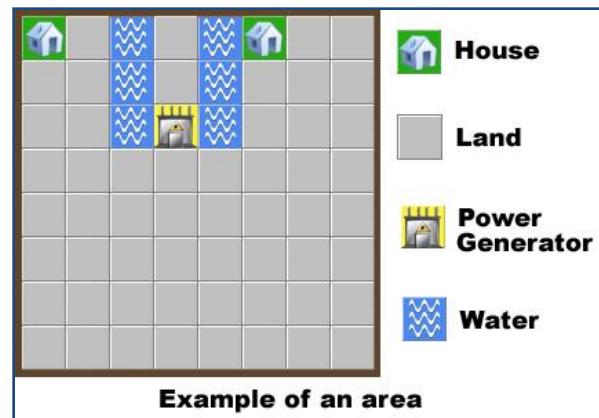
The city 'AjobakahD' has a lot of problems with electricity. Load shedding is a common problem here and people are quite used to it. Instead of calculating the total time the power was on, they calculate the total time the power was off. And of course the later one is always greater.

There is a small area in the city which has not yet been enlightened with any load shedding! That means they haven't got the electricity connection yet. Now the Power Development Board (PDB) wants to set electricity connection in that area. Since the overall power in that city is not sufficient, they have decided to build a power generator in that area and want to connect all the houses to the generator.

The area can be modeled as an **8 x 8** grid. Each cell contains one of the following characters

- '.' means land
- 'H' means house
- 'G' means power generator
- 'W' means water

Two adjacent cells can be connected by cables and the cost is **1** thousand. Two cells are said to be adjacent if they share a side. But two adjacent cells can only be connected if none of the cells is empty. Empty means either land or water. In such case, pillars can be built in the cells and after that they can be connected. The cost of placing a pillar in a land and water cell is **pl** and **pw** thousand respectively. Remember that both the costs can be zero, because there can be sponsors who might use the pillars to advertise themselves.



Now given the modeled grid of the area, the PDB wants to find the minimum cost to connect all the houses to the power generator directly or indirectly. That's why they seek your help as you are one of the finest programmers in town.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with two integers **pl** and **pw** ($0 \leq pl, pw \leq 10$). Then there will be **8** lines, each containing **8** characters from the set $\{'.', 'H', 'G', 'W'\}$. You may assume that in any modeled grid, there is exactly one power generator and the total number of houses is between **1** and **8** (inclusive).

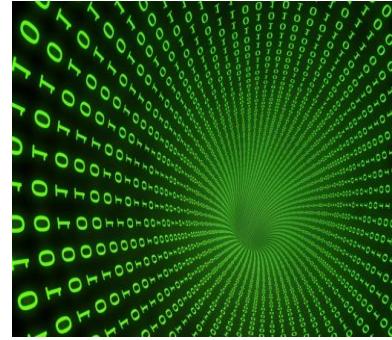
Output

For each test case, print the case number and the minimum total cost in thousands.

Sample Input	Output for Sample Input
<pre>2 0 10 H.W.WH.. ..W.W... ..WGW... 0 0 H.W.WH.. ..W.W... ..WGW...</pre>	<pre>Case 1: 12 Case 2: 7</pre>

1363 - Binary Matrix (II)

You are given an $M \times N$ binary matrix where each cell is either **0** or **1**. This matrix wraps both horizontally and vertically. So, i^{th} row is adjacent to $(i + 1)^{\text{th}}$ row for all $1 \leq i < M$ and M^{th} row is adjacent to 1^{st} row. Similarly, i^{th} column is adjacent to $(i + 1)^{\text{th}}$ column for all $1 \leq i < N$ and N^{th} column is adjacent to 1^{st} column. Obviously row **a** is adjacent to row **b** implies that row **b** is adjacent to row **a**, and same thing is true for columns. Now, two cells of this matrix are adjacent if they are in the same row and their columns are adjacent, or they are in the same column and their rows are adjacent. So for a 3×5 matrix, cell $(2, 3)$ has 4 adjacent cells $(1, 3), (2, 2), (2, 4), (3, 3)$ and cell $(3, 5)$ has 4 adjacent cells $(2, 5), (3, 4), (3, 1), (1, 5)$. Note that, by cell (i, j) we mean the cell of i^{th} row and j^{th} column.



You are only allowed to swap the values of any two adjacent cells of the matrix. Your task is to transform the matrix in such a way that, each of the rows has same number of ones and each of the columns has same number of ones. If it is possible print "**both**". If it is not possible try to make the rows having equal number of ones. If it is possible print "**row**". If it is also not possible try to make the columns having equal number of ones. If it is possible print "**column**". Also print the minimum number of swaps required. But if none of these is possible you have to print "**impossible**".

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with two integers **M** and **N** ($2 \leq M, N \leq 1000$), number of rows and columns of the matrix. Each of the next **M** lines contains **N** characters denoting the rows of the matrix. j^{th} character of the i^{th} line denotes the value of cell (i, j) of the matrix.

Output

For each case, print the case number and the result as specified.

Sample Input	Output for Sample Input
3 2 3 001 111 2 4 0010 0111 2 2 00 01	Case 1: row 1 Case 2: both 3 Case 3: impossible

Note

Dataset is huge, user faster I/O method.

1364 – Expected Cards

Taha has got a standard deck of cards with him. In addition to the 52 regular ones, there are 2 joker cards. Every regular card has a rank and a suit. The ranks in ascending order are: **A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q** and **K**. The suit of a card can be **clubs, diamonds, hearts or spades**. That means there are 13 **clubs**, 13 **diamonds**, 13 **hearts** and 13 **spades** - which adds up to 52. The joker cards have no ranks or suits.

One day, Sara gave Taha a challenge. First she randomly shuffles the 54 cards and starts placing one card after another, face-up, on a table. What is the expected number of cards Sara has to place so that there are at least **C clubs, D diamonds, H hearts** and **S spades** on the table? Whenever a joker card is encountered, Taha has to assign it to some suit so that the expected number of cards to reach the goal is minimized. The decision of assigning the joker card to some suit has to be made instantly (i.e. before Sara puts the next card on the table). Note that the assignments of the two joker cards don't necessarily need to be the same.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a line containing four integers in the order **C, D, H** and **S**. Each of these integers will be in the range **[0, 15]**.

Output

For each case, print the case number first. Then output the expected number of cards Sara needs to place on the table to achieve the goal. If it's impossible to reach the goal, irrespective of what assignments Sara opts for, output **'-1'** (without the quotes) instead. Errors less than **10^{-6}** will be ignored.

Sample Input	Output for Sample Input
4 0 0 0 0 15 13 13 13 1 2 3 4 15 15 15 15	Case 1: 0 Case 2: 54 Case 3: 16.3928186102 Case 4: -1

Notes

1. For case 1, there is no need to place any card as all required values are 0
2. For case 2, we must place all the 54 cards to reach the goal
3. For case 3, note that output isn't always an integer
4. For case 4, 60 Cards? No way!!

1365 – ICPC Guards

This ICPC will take place in a huge hall room which can be divided in to $N \times N$ square cells. That's why some volunteers will guard this room. But each row (or column) should be guarded by exactly two volunteers. And in a single cell at most one volunteer can be placed. Now volunteers can watch other volunteers either vertically or horizontally. Thus the volunteers form different groups. To be more specific, in a single group all the volunteers can look after each other directly or indirectly.

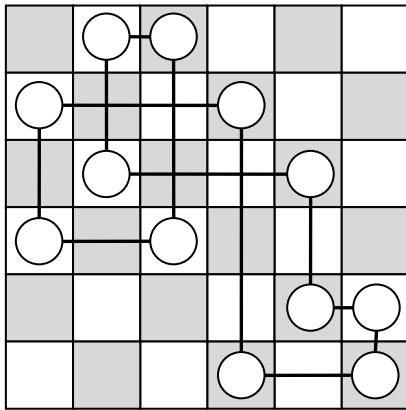


Fig 1

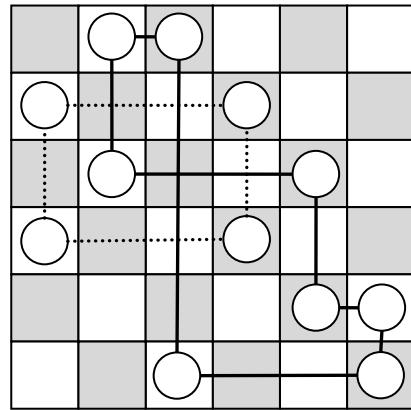


Fig 2

Suppose we have a hall room that can be divided into 6×6 square cells. Circles represent volunteers; lines represent the connectivity of the groups. In Fig 1, there is only one group (check the solid lines carefully). In Fig 2, there are two groups, one group is shown using solid lines, and another one is shown using dotted lines. Now the organizers wanted to know the number of ways they can place volunteers in the hall room such that they form exactly K groups. Two configurations will be different if in one configuration there is a volunteer on a cell but the cell is empty in another one. So, the organizers are seeking your help as you are one of the best programmers in town.

Input

Input starts with an integer T (≤ 50000), denoting the number of test cases.

Each case starts with a line containing two integers: N and K ($2 \leq N \leq 10^5$, $1 \leq K \leq \min(N, 50)$).

Output

For each case, print the case number and the number of ways the volunteers can be placed in the hall room as guards. The result can be large, so print the result modulo **1000 000 007**.

Sample Input	Output for Sample Input
4 2 1 3 1 4 1 4 2	Case 1: 1 Case 2: 6 Case 3: 72 Case 4: 18

1366 – Pair of Touching Circles

You are given a rectangular grid of height \mathbf{H} and width \mathbf{W} . A problem setter wants to draw a pair of circles inside the rectangle so that they touch each other but do not share common area and both the circles are completely inside the rectangle. As the problem setter does not like precision problems, he also wants their centers to be on integer coordinates and their radii should be positive integers as well. How many different ways can he draw such pair of circles? Two drawings are different from each other if any of the circles has different center location or radius.

Input

Input starts with an integer \mathbf{T} (≤ 500), denoting the number of test cases.

Each case starts with a line containing two integers \mathbf{H} and \mathbf{W} ($0 < \mathbf{H}, \mathbf{W} \leq 1000$).

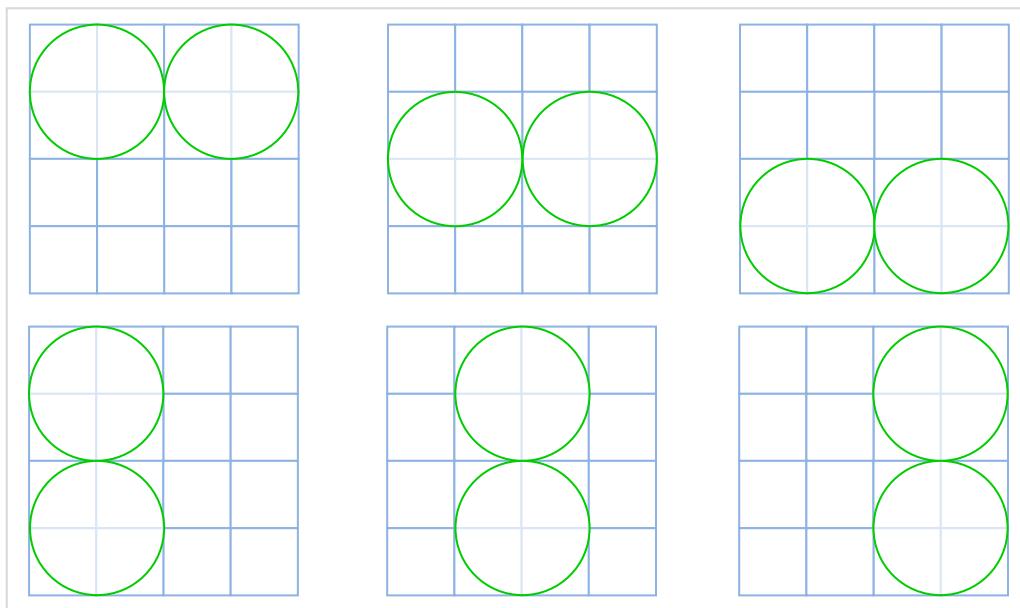
Output

For each case, print the case number and the number of ways of drawing such pairs of circles maintaining the mentioned constraints. Each output will fit into a 64-bit signed integer.

Sample Input	Output for Sample Input
5 4 2 4 3 4 4 4 6 10 10	Case 1: 1 Case 2: 2 Case 3: 6 Case 4: 16 Case 5: 496

Note

For case 3, the possible results are:



1367 – Pizza and Inner Peace

Valley of Peace is not only famous for Inner Peace but also well known for the delicious pizzas of Mr. Ping, the goose father of legendary Kung Fu Panda Po.

The delicious pizza of Mr. Ping has become so much famous that the demand of it has increased in a great extent. Mr. Ping wants to deliver his pizzas to Gongmen City which is ruled by Evil Lord Shen. Wolf bandits are guarding the Gongmen City and everywhere else. So, there are some fixed allowable amounts of pizza C_k that can be carried through k^{th} road. For some reason, the roads are one way only. Assume that there are N cities, Valley of Peace is the 1^{st} city and Gongmen City is the N^{th} city.



Mr. Ping along with Master Fox has calculated the demands for each city and a **least** amount of pizzas L_k ($\leq C_k$) for k^{th} road that must be sent to satisfy the demands. Mr. Ping always wanted Po to become a great chef like him but Panda Po was busy practicing Kung Fu. So, now Mr. Ping has given him the job to deliver pizzas to Gongmen City fulfilling the requirements stated above.

Po along with his team cannot carry more or not even less than allowable amount assigned for each road, otherwise Stealth Mode will be broken and they will get caught. If the amount of pizzas going through the road from i^{th} city to j^{th} ($1 < i, j < N$) city is F_{ij} then for each i , the following condition must hold:

$$\sum_{i=1}^N F_{ij} = \sum_{i=1}^N F_{ji}$$

Panda is very excited to complete this awesome task along with Thundering Rhino and all helping hands possible. As you are a good friend of Panda; he wanted your help to know if it is possible to get this job done.

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers N ($2 \leq N \leq 200$) and M where N denotes the number of cities and M denotes the number of unidirectional roads. The k^{th} line of the next M lines contains four integers u_k v_k L_k C_k ($1 \leq u_k < N$ and $1 < v_k \leq N$, $u_k \neq v_k$, $0 \leq L_k \leq C_k \leq 10^5$) meaning that there is a road from city u_k to v_k , and at least L_k units of pizza must be sent through this road but not exceeding C_k units. There can be at most one road from a city u to v and from any city u , it's impossible to return to u visiting other cities.

Output

For each case, print the case number and "yes" if it's possible to deliver the pizzas maintaining the restrictions, or "no" otherwise. If it's possible to do so, print extra M lines, where the k^{th} line should contain the amount of pizza that should be send through the k^{th} road. There can be multiple solutions, report any valid one.

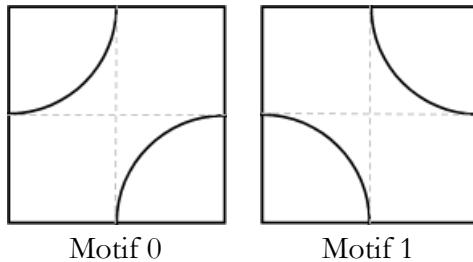
Sample Input	Output for Sample Input
5	Case 1: yes 5
2 1	Case 2: no
1 2 5 10	Case 3: yes 4
4 5	4
1 2 3 4	2
1 3 2 3	2
3 2 2 5	6
3 4 2 3	Case 4: yes
2 4 4 10	5
4 5	5
1 2 3 5	5
1 3 2 10	Case 5: yes
3 2 2 5	5
3 4 2 3	5
2 4 6 10	5
5 4	5
1 4 5 5	0
4 2 3 5	
2 3 3 5	
3 5 0 10	
5 5	
1 4 5 5	
4 2 3 5	
2 3 3 5	
3 5 0 10	
1 5 0 10	

Notes

1. Dataset is huge, use faster I/O methods.
2. This is a special judge problem; wrong output format may cause 'Wrong Answer'.

1368 - Truchet Tiling

In 1704 French mathematician Sébastien Truchet proposed a tiling system that started with simple motifs and used its rotations and random placement to create quite interesting tiling patterns. In this problem we will calculate the area enclosed by the curves in a special case of his tiling system.



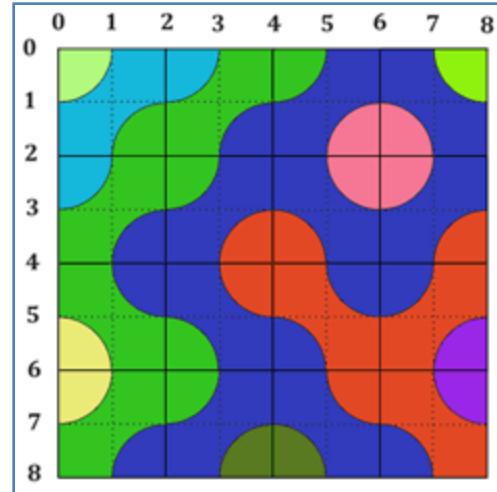
Observe the two motifs here. Motif 0 is a square of length 2 units on each side. There are two circles with radius 1 unit drawn at two opposite corners of the square. Motif 1 is just a 90 degrees rotation of the first one (or you can think of it as drawing the circles on the other two corners).

Using just two basic motifs as shown here, we can tile an area to create rather interesting artistic patterns. They do not have to be symmetrically placed; we can lay out these motifs randomly to cover an area. If we then use a tool like paint bucket (found in most paint programs) at a particular point in the pattern, we can color a contiguous region with one color.

In this problem you'll be given a description of such a tiling pattern and then be asked how much area would it color if we use paint bucket starting at specific points.

Here is one illustration:

We have the motifs randomly placed on a 9×9 grid. Then we used a blue color at position $(2, 4)$ (here 2 denotes the distance from the top and 4 denotes the distance from left). It colored the entire contiguous region it found bounded by the grid's boundary and the curves themselves. We could achieve the same effect if we used the color at position $(4, 6)$, $(0, 6)$, $(1, 7)$ etc. However, using the paint bucket on the perimeter of the curves (such as $(3, 4)$) will only color the perimeter line of the curves. They will not fill up a region with any color.



Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers R and C ($1 \leq R, C \leq 100$). Each of the next R lines contains the row description given in binary form. The basic motifs are numbered **0** and **1**. For example, the accompanying picture's first row can be described as "**0001**".

The next line contains an integer **Q** ($1 \leq Q \leq 100$) denoting the number of queries. Each query will be of the form **x y** ($0 \leq x \leq 2R$, $0 \leq y \leq 2C$) and where **x** and **y** are integers denoting the row and column number where the paint bucket tool will be used.

Output

For each case, print the case number in a single line. Then for each query, print one number giving the area enclosed by the boundary and the curves that contain that point. If the point in question itself lies on a curve, you can assume the enclosed area to be zero. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
3 1 2 01 4 0 0 2 0 0 1 0 2 2 2 01 00 1 2 2 3 1 1 0 1 2 3 1 4 2	Case 1: 0.7853981634 4.8584073464 0 4.8584073464 Case 2: 4.7853981634 Case 3: 7.2876110196 1.5707963268

Note

This is a special judge problem; wrong output format may cause 'wrong answer'.

1369 – Answering Queries

The problem you need to solve here is pretty simple. You are given a function $f(A, n)$, where A is an array of integers and n is the number of elements in the array. $f(A, n)$ is defined as follows:

```
long long f( int A[], int n ) { // n = size of A
    long long sum = 0;
    for( int i = 0; i < n; i++ )
        for( int j = i + 1; j < n; j++ )
            sum += A[i] - A[j];
    return sum;
}
```

Given the array A and an integer n , and some queries of the form:

- 1) $0 \leq x < n, 0 \leq v \leq 10^6$, meaning that you have to change the value of $A[x]$ to v .
- 2) 1, meaning that you have to find f as described above.

Input

Input starts with an integer T (≤ 5), denoting the number of test cases.

Each case starts with a line containing two integers: n and q ($1 \leq n, q \leq 10^5$). The next line contains n space separated integers between 0 and 10^6 denoting the array A as described above.

Each of the next q lines contains one query as described above.

Output

For each case, print the case number in a single line first. Then for each query-type "1" print one single line containing the value of $f(A, n)$.

Sample Input	Output for Sample Input
1 3 5 1 2 3 1 0 0 3 1 0 2 1 1	Case 1: -4 0 4

Note

Dataset is huge, use faster I/O methods.

1370 - Bi-shoe and Phi-shoe

Bamboo Pole-vault is a massively popular sport in Xzhiland. And Master Phi-shoe is a very popular coach for his success. He needs some bamboos for his students, so he asked his assistant Bi-Shoe to go to the market and buy them. Plenty of Bamboos of all possible integer lengths (yes!) are available in the market. According to Xzhila tradition,

Score of a bamboo = **Φ (bamboo's length)**

(Xzhilans are really fond of number theory). For your information, **Φ (n)** = numbers less than **n** which are relatively prime (having no common divisor other than 1) to **n**. So, score of a bamboo of length 9 is 6 as 1, 2, 4, 5, 7, 8 are relatively prime to 9.

The assistant Bi-shoe has to buy one bamboo for each student. As a twist, each pole-vault student of Phi-shoe has a lucky number. Bi-shoe wants to buy bamboos such that each of them gets a bamboo with a score greater than or equal to his/her lucky number. Bi-shoe wants to minimize the total amount of money spent for buying the bamboos. One unit of bamboo costs 1 Xukha. Help him.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 10000$)** denoting the number of students of Phi-shoe. The next line contains **n** space separated integers denoting the lucky numbers for the students. Each lucky number will lie in the range **[1, 10^6]**.

Output

For each case, print the case number and the minimum possible money spent for buying the bamboos. See the samples for details.

Sample Input	Output for Sample Input
3 5 1 2 3 4 5 6 10 11 12 13 14 15 2 1 1	Case 1: 22 Xukha Case 2: 88 Xukha Case 3: 4 Xukha

1371 – Energetic Pandas

There are **n** bamboos of different weights W_i . There are **n** pandas of different capacity CAP_i . How many ways the pandas can carry the bamboos so that each panda carries exactly one bamboo, every bamboo is carried by one panda and a panda cannot carry a bamboo that is heavier than its capacity. Two ways will be considered different if at least one panda carries a different bamboo.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 1000$) denoting the number of pandas and bamboos. The next line contains **n** space separated distinct integers denoting the weights of the bamboos. The next line contains **n** space separated distinct integers denoting the capacities for the pandas. The weights and the capacities lie in the range $[1, 10^9]$.

Output

For each case, print the case number and the number of ways those pandas can carry the bamboos. This number can be very big. So print the result modulo **1000 000 007**.

Sample Input	Output for Sample Input
3 5 1 2 3 4 5 1 2 3 4 5 2 1 3 2 2 3 2 3 4 6 3 5	Case 1: 1 Case 2: 0 Case 3: 4

1372 - Hexagonal Bamboo Fort

You are given the lengths of **N** bamboo sticks that have distinct lengths. You have to select six sticks from those to make a fort of convex hexagonal shape. The hexagon should have a strictly positive area. For example, the sticks $\{1, 2, 3, 4, 5, 6\}$ can be used to make a valid hexagon but the sticks $\{1, 2, 3, 4, 5, 20\}$ can't make any valid hexagon with positive area.

How many ways you can do that? Consider that, two hexagons will be different, if the set of their sides are not same. For example, the hexagon with sides $\{1, 2, 3, 4, 5, 6\}$ and $\{1, 3, 5, 4, 2, 6\}$ are same but $\{1, 2, 3, 4, 5, 6\}$ and $\{1, 2, 3, 4, 5, 7\}$ are not.

Input

Input starts with an integer **T** (≤ 25), denoting the number of test cases.

Each case starts with a line containing an integer **N** ($6 \leq N \leq 100$). Next line contains **N** distinct integers giving the lengths of the sticks. Each of them will be between **1** and **10^5** (inclusive).

Output

For each case, print the case number and the number of ways six sticks can be selected to make a valid convex hexagon with positive area.

Sample Input	Output for Sample Input
3 6 1 2 3 4 5 6 8 1 2 5 20 30 10 100 400 8 179 259 263 265 506 146 222 828	Case 1: 1 Case 2: 1 Case 3: 28

1373 – Strongly Connected Chemicals

One type of chemical compounds is the ionic compound, because it is composed of positively charged ions (cations) and negatively charged ions (anions) in such a way that the compound as a whole is electrically neutral. Cations always attract anions and anions always attract cations.

Now you are given **m** cations and **n** anions, and their connectivity information about which cation attracts which anion. Your task is to find a sub group from the cations and anions where there is at least one cation and at least one anion, such that all the cations in the group attract all the anions in the group. We call such a group as '**Strongly Connected Chemicals**'. As there can be many such groups, we want to find a group which has the highest cardinality. Cardinality means the number of members (cation and anion) in the group.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing two integers: **m** and **n** ($1 \leq m, n \leq 50$). Each of the next **m** lines contains **n** characters (without space), each either '0' or '1'. If the **jth** character in the **ith** line is 1, that means the **ith** cation attracts the **jth** anion, otherwise it doesn't.

Output

For each case, print the case number and the maximum possible size of the strongly connected chemical group.

Sample Input	Output for Sample Input
2 3 4 1100 1110 0011 2 2 10 00	Case 1: 4 Case 2: 2

1374 – Confusion in the Problemset

A small confusion in a problem set may ruin the whole contest. So, most of the problem setters try their best to remove any kind of ambiguity from the set. But sometimes it is not that important. For example, the mock contest of ICPC Dhaka Regional. As it is mock contest so we are not that serious with the set. We printed two problems, problem **A** in Page 1 and Problem **B** in Page 2. Then we remembered that we had to give rule of the contest too. Thus we printed the rule page. But we did not notice that the rule page was printed with Page 2. We were stapling 3 pages together. First rule page, then Problem **A** and at the last Problem **B**. So, the written page numbers were, 2, 1 and 2. This looked odd. But we already printed all the pages and if we want to fix the issue we had no other way but to print all the three pages. One among us suggested an explanation, "Well, first 2 means there are 2 pages after this page. 1 also means there is 1 page after this page. But the 2 in last page means there are 2 pages before this page." Interesting observation indeed! So we came up with a rule which is, page numberings of all the **n** pages are valid, if the page number at a page denotes number of page before this page or number of page after this page.

So with this rule, {3, 1, 2, 0} is valid but {3, 3, 1, 3} is not valid.

Input

Input starts with an integer **T** (≤ 60), denoting the number of test cases.

Each case starts with a line an integer **n** ($1 \leq n \leq 10000$) denoting the number of pages in the problem-set. The next line contains **n** space separated integers denoting the page number written on the pages. The integers lie in the range [0, 10^6].

Output

For each case, print the case number and "yes" if the pages can be shuffled somehow to meet the given restrictions. Otherwise print "no".

Sample Input	Output for Sample Input
2 4 0 3 1 2 4 1 3 3 3	Case 1: yes Case 2: no

Notes

1. For case 1, the pages can be shuffled in several ways so that the page numbering is valid.
One of the valid shuffles is 3, 1, 2, 0.
2. For case 2, there is no valid way to shuffle these.

1375 - LCM Extreme

Find the result of the following code:

```
unsigned long long allPairLcm( int n ) {
    unsigned long long res = 0;
    for( int i = 1; i <= n; i++ )
        for( int j = i + 1; j <= n; j++ )
            res += lcm(i, j); // lcm means least common multiple
    return res;
}
```

A straight forward implementation of the code may time out.

Input

Input starts with an integer T ($\leq 2 \times 10^5$), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 3 \times 10^6$).

Output

For each case, print the value returned by the function '`allPairLcm(n)`'. As the result can be large, we want the result modulo 2^{64} .

Sample Input	Output for Sample Input
4	Case 1: 2
2	Case 2: 1036
10	Case 3: 3111
13	Case 4: 9134672774499923824
100000	

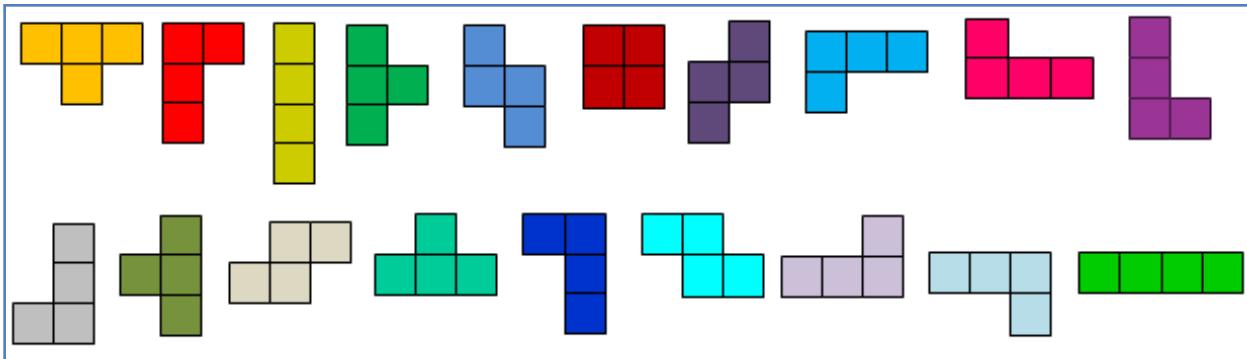
Note

Dataset is huge, use faster I/O Methods.

1376 – Tetromino

Dexter wants to completely cover a $4 \times N$ board using N tetrominoes. Every cell in the grid has to be covered by exactly one piece and no piece is allowed to go outside the board. And no piece can be rotated or flipped.

A tetromino is a connected set of 4 tiles. All 19 possible tetrominoes that can be used (any number of times) are shown below:



Assume that all 19 pieces have different colors. Now your task is to find the total number of ways Dexter can cover the board. Two board configurations are different if a cell can be found where the colors are different.

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with a line containing an integer N ($1 \leq N \leq 10^9$).

Output

For each case, print the case number and the number of ways to fill the board modulo **1000 000 007**.

Sample Input	Output for Sample Input
4	Case 1: 1
1	Case 2: 4
2	Case 3: 23
3	Case 4: 15747348
12	

1377 – Blade and Sword

You may have played the game 'Blade and Sword', it's an action game. However, in this problem we are actually solving one stage of the game.

For simplicity, assume that the game stage can be modeled as a **2D** grid which has **m** rows and **n** columns. The cells are categorized as follows:

- 1) '.' represents an empty space. The player can move through it.
- 2) '#' represents wall, and the player cannot move through it. You can assume that the boundaries of the grid will be walls.
- 3) '*' means a teleporting cell, once the player moves into this cell, he must choose **any other** teleporting cell where he will be taken to. But if he cannot find a desired teleporting cell, he will die. However, after moving to the desired teleporting cell, he can either move to an adjacent cell, or he can teleport again using the same procedure.
- 4) 'P' means the position of the player and there will be exactly one cell containing 'P'.
- 5) 'D' means the destination cell and there will be exactly one cell containing 'D'.

Now you are given a stage and the player starts moving. It takes one unit of time for the player to move to any adjacent cell from his current position. Two cells are adjacent if they share a side. One unit of time is needed for the teleporting service; that means taking the player from one teleporting cell to any other teleporting cell. Your task is to find the minimum possible time unit required for the player to reach the destination cell.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers: **m** and **n ($3 \leq m, n \leq 200$)**. Each of the next **m** lines contains **n** characters denoting the stage. The given stage follows the restrictions stated above.

Output

For each case, print the case number and the minimum required time. If it's impossible for the player to reach the destination cell, print '**impossible**'. See the samples for details.

Sample Input	Output for Sample Input
2 4 10 ##### #.P..#*..# #*.....D# ##### 3 9 ##### #P.#..D.# #####	Case 1: 6 Case 2: impossible

1378 – The Falling Circle

The little Jerry is now developing a game. You know developing game is a complex thing as there are many challenges involved. For example, little Jerry is now stuck with some geometric pattern and asks for your help. To help him you do not need to know every detail of the game, only information of the particular frame should suffice. Here it is:

Two circles are attached with the wall. They are fixed. A line is attached to the circles in such a way that it touches both the circles and each of the touching points has lower **Y** coordinate than that of the corresponding center of the circle. Now another circle is dropped on the set up from above (higher **Y** coordinate). The circle will fall along the **Y** axis with a constant velocity of **1** unit per second until it touches the line. When it touches the line, it starts to rotate along the line towards the circle that has lower **Y** coordinate touching point with a constant angular velocity of **1** revolution per second. When it touches the circle at the end, it stops. If both the touching points have same **Y** coordinate, i.e. the line is parallel to the **X** axis, then the falling circle stops as soon as it touches the line. Now given the setup, you need to find the time after which the falling circle will stop.

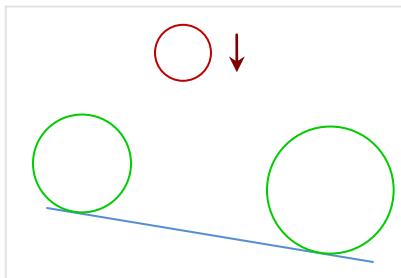


Figure 1: Initial Setup, the circle starts falling

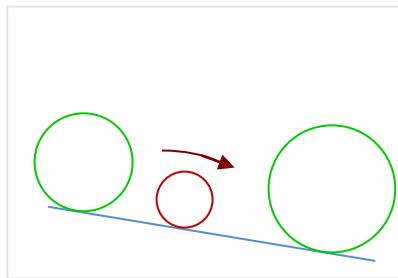


Figure 2: The circle touches the line and starts to rotate

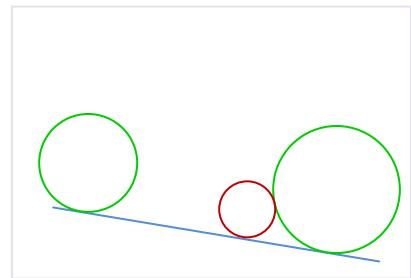


Figure 3: The circle touches the circle at the end and stops

Input

Input starts with an integer **T** (≤ 10000), denoting the number of test cases.

Each case contains three lines. First two lines contain information of the fixed circles on the line and the 3rd line describes the falling circle. Each of the circles are described with three integers **x**, **y** and **r** ($-10^5 \leq x, y \leq 10^5$, $0 < r \leq 100$), where (x, y) denotes the co-ordinate of the center, **r** denotes the radius. The falling circle will always touch the line first and will drop between the fixed circles.

Output

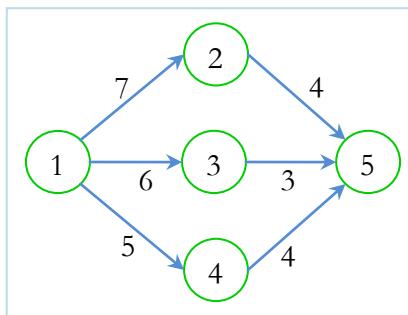
For each case, print the case number and the required time in seconds. Error less than 10^{-6} will be ignored. Assume that the falling circle will not bounce when it drops on the line.

Sample Input	Output for Sample Input
1 0 0 5 100 1 6 50 60 10	Case 1: 55.000000

1379 – Toll Management

In Dhaka there are too many vehicles. So, the result is well known, yes, traffic jam. So, mostly people have to spend quite a time in the roads to go from one place to another.

Now, the students have finally found a solution to this problem. The idea is to make all the roads one way. That means a vehicle can go through the roads in one way only. And to make the number of vehicles low, each vehicle has to pay a toll to use a road. Now you want to go from a place s to another place t . And you have a total of p taka in your pocket. Now you want to find the path which contains the highest toll road, to go from s to t . Remember that you can't use more than p taka.



For the given picture, $s = 1$, $t = 5$ and $p = 10$. There are three paths from 1 to 5.

1. Path 1: $1 - 2 - 5$, total toll = 11 ($> p$)
2. Path 2: $1 - 3 - 5$, total toll = 9 ($\leq p$), 6 is the maximum toll
3. Path 3: $1 - 4 - 5$, total toll = 9 ($\leq p$), 5 is the maximum toll

So the maximum toll for a road of all of the paths having total toll not greater than p is 6.

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case starts with five integers N ($2 \leq N \leq 10000$), M ($1 \leq M \leq 50000$), s ($1 \leq s \leq N$), t ($1 \leq t \leq N$) and p ($1 \leq p \leq 10^6$) where N means the number of junctions and M means the number of roads connecting the junctions. Then there will be M lines each containing three integers u v c . u and v are junctions and there is a road from u to v ($1 \leq u, v \leq N$, $u \neq v$) and c ($0 \leq c \leq 10^5$) is the toll needed for that road. There can be multiple roads between two junctions.

Output

For each case, print the case number and the desired result. If no such result is found, print "-1".

Sample Input	Output for Sample Input
2 5 6 1 5 10 1 2 7 2 5 4 1 3 6 3 5 3 1 4 5 4 5 4 2 1 1 2 10 1 2 20	Case 1: 6 Case 2: -1

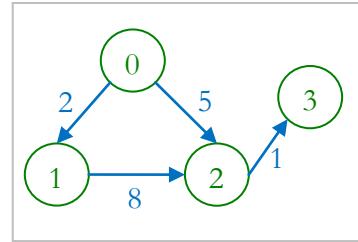
Note

Dataset is huge, use faster I/O methods.

1380 – Teleport

You are in a tour and want to visit all the n cities in a country which are numbered from 0 to $n - 1$. Initially you are standing on city K . You have a car and can drive to one city to another. And you have a teleporting device, too. It can take you to any place to another place in no time. But the problem is that, the teleporting machine only works if the place you want to go is already visited by you. It can't detect the co-ordinates of a new city. But when you visit a city, the device locates the co-ordinates carefully, and thus helps you to teleport to this city from any other place. And the device is so strong that you can even teleport using the device along with your car! Since the traffic of the country is so high, the Govt. has planned to use only m unidirectional roads. Now you are given the map of the city and the required times for you to drive the roads. You have to find the minimum possible time needed for you to visit all the cities.

For example, for the city in the picture, you are in city 0 . At first you go to city 1 , and it will take 2 units of time, then you can go to city 2 from city 1 , but it will take 8 units of time. You can teleport to city 0 and then go to city 2 , it will cost you 5 . So, you can visit city 1 and 2 in 7 units of time. And you can visit all the cities in time unit 8 .



Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a black line. Next line contains three integers: $n \ m \ K$ ($1 \leq n \leq 1000, 0 \leq m \leq 10000, 0 \leq K < n$). Each of the next m lines contains three integers $u \ v \ w$ ($0 \leq u, v < n, u \neq v, 1 \leq w \leq 10000$) meaning that there is a road from city u to city v and takes w units of time to use that road. There can be at most one road from a city u to another city v .

Output

For each case, print the case number and the minimum time needed to visit all the cities. If it's impossible to do so, print "impossible".

Sample Input	Output for Sample Input
2 4 4 0 0 1 2 1 2 8 0 2 5 2 3 1 2 1 1 0 1 10	Case 1: 8 Case 2: impossible

Note

Dataset is Huge, use faster I/O Methods.

1381 - Scientific Experiment

John wants to be a scientist. A first step of becoming a scientist is to perform experiment. John has decided to experiment with eggs. He wants to compare the hardness of eggs from different species. He has decided to use a nearby large multi-storied building for this purpose. For each species he will try to find the highest floor from which he can drop the egg and it will not break. The building has $(n+1)$ floors numbered from **0** to **n**. John has a book from which he knows that

1. If an egg is dropped from the topmost floor, it will surely break.
2. If an egg is dropped from floor **0**, it will not break.
3. The eggs of same species are of same strength. That means if any egg breaks when dropped from the **kth** floor; all the eggs of that species will break if dropped from **kth** floor.
4. If an egg is unbroken after dropping it from any floor, it remains unharmed, that means the strength of the egg remains same.



Unfortunately John has a few problems:

1. He can only carry one egg at a time.
2. He can buy eggs from a shop inside the building and an egg costs **x** cents.
3. To enter the building he has to pay **y** cents if he has no egg with him and **z** cents if he carries an egg with him.
4. After dropping an egg, John must go outside the building to check whether it's broken or not.
5. He does not want to waste any egg so he will not leave any unbroken egg on the ground. But if an egg is broken, he leaves it there.
6. If he has an intact egg at the end, he can sell it for **x/2** cents. He does not need to enter the building to sell the egg.

These problems are not going to tame John's curious mind. So he has decided to use an optimal strategy and minimize his cost in worst case. As John is not a programmer, he asked your help.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a line containing four integers **n x y z** as described in the statement. You may assume that **1 < n ≤ 1000** and **1 $\leq x, y, z \leq 10^5$** and **x** is even.

Output

For each test case, print the case number and the minimized worst case cost.

Sample Input	Output for Sample Input
7	Case 1: 2000
4 2 998 1000	Case 2: 4008
16 2 1000 1000	Case 3: 1015
16 1000 1 1	Case 4: 1003
4 1000 1 1	Case 5: 10
7 2 2 2	Case 6: 24
9 2 1 100	Case 7: 111
11 2 100 1	

Note

For case 1, John knows that the egg will break if dropped from 4th floor, but will not break if dropped from 0th floor. An optimal solution may be

1. John enters the building without any egg (¢998).
2. John buys an egg (¢2).
3. John drops an egg from 2nd floor. John goes out and checks the egg.
 - a. If it breaks,
 - i. John again enters the building without any egg (¢998) and buys an egg there ¢2.
 - ii. He drops the egg from 1st floor.
 1. If it does not break then answer to his problem is 1 and he can sell the egg for ¢1. So his final cost in ¢1999.
 2. If it breaks then the answer to his problem is 0th floor and his final cost is ¢2000.
 - b. If it does not break
 - i. John enters the building with the egg (¢1000).
 - ii. He drops it from 3rd floor.
 1. If it does not break then answer to his problem is 3 and he can sell the egg for ¢1. So his final cost in ¢1999.
 2. If it breaks then the answer to his problem is 2 and final cost is ¢2000.

So, using this strategy, his worst case cost is ¢2000.

1382 – The Queue

On some special occasions Nadia's company provide very special lunch for all employees of the company. Before the food is served all of the employees must stand in a queue in front of the food counter. The company applied a rule for standing in the queue. The rule is nobody can stand anywhere in front of his supervisor in the queue. For example, if Abul is the supervisor of Babul and Abul stands in k^{th} position from the front of the queue, then Babul cannot stand at any position in between 1 and $k - 1$ from front of the queue.

The company has N employees and each of them has exactly one supervisor except one (CEO) who doesn't have any supervisor.

You have to calculate in how many ways the queue can be created. For this problem, you can safely assume that in at least one way the queue can be created.

Input

Input starts with an integer T (≤ 700), denoting the number of test cases.

Each case starts with a line containing an integer N ($1 \leq N \leq 1000$). Each of the following $N - 1$ lines will contain two integers a and b ($1 \leq a, b \leq N, a \neq b$), which denotes that a is the supervisor of b . For the sake of simplicity we are representing each employee by an integer number. Assume that the given input follows the restrictions stated above.

Output

For each case, print the case number and the number of ways to create the queue. The result can be large, print the result modulo **1000 000 007**.

Sample Input	Output for Sample Input
1 5 2 1 2 3 3 4 3 5	Case 1: 8

1383 – Underwater Snipers

King **Motashota** is in a war against the mighty **Bachchaloks**. He has formed a well trained army of snipers, and planning to use them as much as possible. In one of the missions, he has **S** snipers. They will be dispatched to get rid of the soldiers guarding the bank of the river **Nodi**.

From satellite images, **Motashota** has located positions of all enemy soldiers. Now, the plan is, snipers will take their positions. They are excellent swimmers, so, you can assume that they won't get caught, while taking position. Upon order from **Motashota**, they will start shooting enemy soldiers. A sniper can shoot a soldier, if Euclidean Distance between the soldier and sniper is no more than **D**. After the snipers get rid of all the soldiers, they can proceed with the operation. So, it is important for them to position the snipers in such a way that, all soldiers are within the range of at least one sniper.

In addition, when snipers start shooting, the guards will be alert, and thus, snipers can't change their position, they can only continue shooting from their position.

The river bank is defined by the horizontal line $y = k$. All points (x, y) where $y > k$ is in the enemy territory, and if $y < k$, then it's on the water. You will be given location of **N** soldiers, strictly in the enemy territory; you have to place **S** snipers in the water, so that they can kill all soldiers. For security reasons, the snipers should be as far from the bank as possible. For any sniper in position (x_i, y_i) , the distance from the bank is $|y_i - k|$. If, for all snipers, the minimum of them is $M = \min\{|y_i - k|\}$, you have to maximize **M**.

Both the soldiers and snipers are really superstitious. They will stay only in integer coordinates.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a black line. Next line contains four integers, **k** ($-10^8 \leq k \leq 10^8$), **N** ($1 \leq N \leq 10000$), **S** ($1 \leq S \leq 10000$) and **D** ($1 \leq D \leq 10^9$), the position of the bank, number of guards and number of snipers, and the range of the snipers.

This is followed by **N** lines, each containing a pair of integers (x_i, y_i) the position of i^{th} guard ($-10^8 \leq x_i \leq 10^8$, $k < y_i \leq 10^8$).

Output

For each case, print the case number and **M** which is described in the statement. If the snipers cannot kill all the guards, print "**impossible**".

Sample Input	Output for Sample Input
<p>2</p> <p>0 3 2 4 1 1 3 2 9 1</p> <p>0 3 1 4 1 1 3 2 9 1</p>	<p>Case 1: 2 Case 2: impossible</p>

1384 – Stream My Contest

During 2009 and 2010 ICPC world finals, the contests were webcasted via World Wide Web. Seeing this, some contest organizers from Ajobdesh decided that, they will provide a live stream of their contests to every university in Ajobdesh. The organizers have decided that, they will provide best possible service to them. But there are two problems:

1. There is no existing network between universities. So, they need to build a new network. However, the maximum amount they can spend on building the network is **C**.
2. Each link in the network has a bandwidth. If, the stream's bandwidth exceeds any of the link's available bandwidth, the viewers, connected through that link can't view the stream.

Due to the protocols used for streaming, a viewer can receive stream from exactly one other user (or the server, where the contest is organized). That is, if you have two 128kbps links, you won't get 256kbps bandwidth, although, if you have a stream of 128kbps, you can stream to any number of users at that bandwidth.

Given **C**, find the maximum possible bandwidth they can stream.

Input

Input starts with an integer **T** (≤ 35), denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers **N, M, C** ($1 \leq N \leq 60, 1 \leq M \leq 10^4, 1 \leq C \leq 10^9$), the number of universities and the number of possible links, and the budget for setting up the network respectively. Each university is identified by an integer between **0** and **N-1**, where **0** denotes the server.

Each of the next **M** lines contains four integers **u, v, b, c** ($0 \leq u, v < N, 1 \leq b, c \leq 10^6$), describing a possible link from university **u** to university **v**, that has the bandwidth of **b** kbps and of cost **c**. All links are unidirectional. There can be multiple links between two universities.

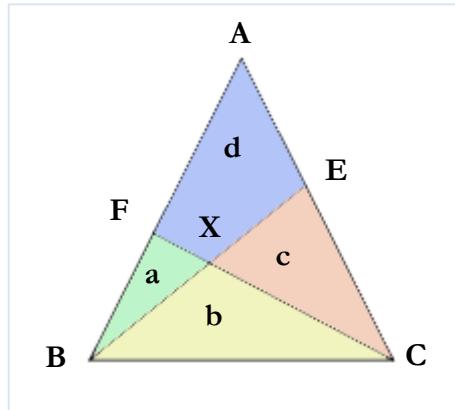
Output

For each case, print the case number and the maximum possible bandwidth to stream. If it's not possible, print "**impossible**". See the samples for details.

Sample Input	Output for Sample Input
<pre> 3 3 4 300 0 1 128 100 1 2 256 200 2 1 256 200 0 2 512 300 3 4 500 0 1 128 100 1 2 256 200 2 1 256 200 0 2 512 300 3 4 100 0 1 128 100 1 2 256 200 2 1 256 200 0 2 512 300 </pre>	Case 1: 128 kbps Case 2: 256 kbps Case 3: impossible

1385 - Kingdom Division

The king of **Geometry-Land** was in deep trouble. His three sons quarrel all the time. The king tried a lot but in vain. "How about dividing the kingdom?" the king thought to himself. So, he called the advisors and described his plan. They opened the map...



The kingdom is triangular. The king denoted the three vertices as **A**, **B**, **C**. He drew a line from **B** to **E** (**E** is any point in segment **AC**) and another line from **C** to **F** (**F** is any point in segment **AB**). The intersection of **BE** and **CF** was denoted by **X**.

Then they got four parts - **a** (triangle **BFX**), **b** (triangle **BCX**), **c** (triangle **CEX**) and **d** (quadrilateral **AEXF**). The king decided to give these areas - **a**, **b**, **c** to his three sons. And the area **d** would be the king's new kingdom.

Now, you are given the value of **a**, **b** and **c**. You have to find the area of **d**.

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case starts with a line containing three positive real numbers **a** **b** and **c** which are not greater than **1000**. You can also assume that the numbers contain at most four places after the decimal point.

Output

For each case, print the case number first. Then if the area of **d** cannot be determined, print "**-1**". Otherwise print the area of **d**. Errors less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 1 2 1 2 4 2 1 3 3 1.28 2.67 3.12	Case 1: 2 Case 2: 4 Case 3: 5 Case 4: 12.4063611138

1386 - Jaguar King

In a deep forest, a war is going to begin. Like other animals, the jaguars are preparing for this ultimate battle. Though they are mighty strong and lightening fast, they have an extra advantage over other animals. It's their wise and brave king - 'The Jaguar King'.

The king knows that only speed and strength is not enough for winning the war. They have to make a perfect formation. The king has set up a nice formation and placed all the jaguars according to that formation. There are N jaguars (including the king). The king is marked by 1 and the other jaguars are marked by a number from 2 to N . There are N positions numbered from 1 to N , and initially the jaguars are positioned as their number i.e. the i^{th} jaguar is in i^{th} position.



But then the king realizes that to make the formation perfect and effective, some positions should have stronger jaguars and some should have faster jaguars. Since the strength and speed of all the jaguars are not same, the king decided to change the positions of some jaguars. The wise king knows the ability of each and every jaguar, so his decision is perfect but the problem is how to change the position of the jaguars.

One of the wise jaguars has given an idea. The idea is simple. All the jaguars will wait for the king's signal, all eyes upon the king. Suppose the king is in the i^{th} position. The king jumps to j^{th} position and when the jaguar at j^{th} position sees the king coming, he immediately jumps to i^{th} position. The king repeats this procedure until they are formatted like the new formation. And collision will never occur in the jumping procedure.

If the king is in the i^{th} position,

- if i modulo 4 is 1, the king can jump to position $(i+1), (i+3), (i+4), (i-4)$
- if i modulo 4 is 2, the king can jump to position $(i+1), (i-1), (i+4), (i-4)$
- if i modulo 4 is 3, the king can jump to position $(i+1), (i-1), (i+4), (i-4)$
- if i modulo 4 is 0, the king can jump to position $(i-3), (i-1), (i+4), (i-4)$

And assume that the king always jumps to a valid position i.e. position between 1 and N .

Now you are given the final formation of the jaguars, your target is to find the minimum number of times the king has to jump to gain the new formation.

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a line containing an integer N ($4 \leq N \leq 20$, N is a multiple of 4). The next line contains N integers, which is a permutation of 1 to N , denoting the final formation of the jaguars. The i^{th} integer denotes the jaguar which should be placed to i^{th} position.

Output

For each case, print the case number first. Then if it's impossible to do so, print '**impossible**'. If it's possible to do so, but not with in 25 jumps, print '**impossible in 25 jumps**'. Otherwise, print the minimum number of times the king has to jump to gain the new formation.

Sample Input	Output for Sample Input
8 4 1 2 3 4 4 4 2 3 1 8 5 2 3 4 8 6 7 1 8 5 2 8 3 6 7 1 4 4 4 1 3 2 8 8 1 4 3 6 7 5 2 12 12 2 4 1 10 9 7 8 5 6 3 11 12 1 3 12 10 4 6 7 8 5 9 2 11	Case 1: 0 Case 2: 1 Case 3: 2 Case 4: 7 Case 5: impossible Case 6: 15 Case 7: 25 Case 8: impossible in 25 jumps

1387 - Setu

Rahaduzzaman Setu, (Roll - 12) of 13th batch, CSE, University of Dhaka. He passed away on 18th April 2012. This is one of the saddest news to all. May he rest in peace. This problem is dedicated to him.

This problem was written during his treatment. He will be in our prayers, always.

"He has been suffering from Multi Drug Resistant TB for a long time. Now, his left lung is damaged and beyond repair. No medicine is working on his body to ease his pain. It is urgent to operate on his left lung so that the disease doesn't spread to his right lung. It can either be removed through surgery or transplanted. He comes from a modest family and it is difficult and impossible for them to bare his medical expenses anymore. Because of the money needed (12 million BDT) to transplant, it is his family's decision to go with the surgery (3 million BDT). We must help them financially by raising money. But we must not be confined with that amount only to do the surgery. We must go for the Transplant. Our target will be to collect as much as possible to help our friend [\[link\]](#)."

However, in this problem, you have to build a software that can calculate the donations. Initially the total amount of money is 0 and in each time, two types of operations will be there.

- 1) "donate K" ($100 \leq K \leq 10^5$), then you have to add K to the account.
- 2) "report", report all the money currently in the account.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **N** ($1 \leq N \leq 100$) denoting the number of operations. Then there will be **N** lines each containing two types of operations as given. You may assume that the input follows the restrictions above. Initially the account is empty for each case.

Output

For each case, print the case number in a single line. Then for each "report" operation, print the total amount of money in the account in a single line.

Sample Input	Output for Sample Input
2 4 donate 1000 report donate 500 report 2 donate 10000 report	Case 1: 1000 1500 Case 2: 10000

1388 – Trapezium Drawing

Little Jerry is now in a trouble. The teacher gave him a task. He needs to draw a trapezium from the lengths of the sides. Little Jerry has started the drawing and he has drawn the bigger parallel side but cannot proceed from here. So he asks for your help. As he has already got two points of the trapezium, you just need to tell him the other two points so that he can complete his drawing. More specifically, consider a Trapezium **ABCD** where the points are in counter clockwise order and **AB** is the larger parallel side. You are given point **A**, point **B** and the lengths **BC**, **CD**, **DA**. You need to find points **C** and **D**. See the picture below:

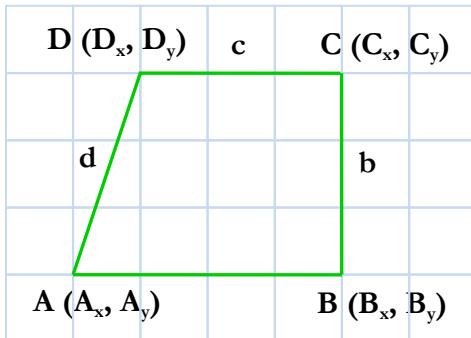


Fig: a trapezium

Input

Input starts with an integer **T** (≤ 500), denoting the number of test cases.

Each case starts with a line containing seven integers **A_x**, **A_y**, **B_x**, **B_y**, **b**, **c**, **d**. Here **(A_x, A_y)** is the coordinate of point **A**, **(B_x, B_y)** is the coordinate of point **B**. **b**, **c** and **d** are the lengths of the segments **BC**, **CD** and **DA** respectively. All the coordinates will be in the range **[-10000, 10000]** and the lengths will be in the range **[1, 10000]**. You may assume that a valid trapezium is possible with the given data and the length of **CD** will be strictly less than the length of **AB**.

Output

For each line of input produce two lines of output. The first line will contain the case number and the second line will contain four real numbers **C_x**, **C_y**, **D_x**, **D_y** where **(C_x, C_y)** is the coordinate of point **C** and **(D_x, D_y)** is the coordinate of point **D**. Errors less than **10⁻⁶** will be ignored.

Sample Input	Output for Sample Input
2 0 0 20 0 10 14 8 0 0 14 0 7 11 5	Case 1: 14 8 0 8.000 Case 2: 8.5 4.33012701 -2.5 4.33012701

1389 – Scarecrow

Taso owns a very long field. He plans to grow different types of crops in the upcoming growing season. The area, however, is full of crows and Taso fears that they might feed on most of the crops. For this reason, he has decided to place some scarecrows at different locations of the field.

The field can be modeled as a $1 \times N$ grid. Some parts of the field are infertile and that means you cannot grow any crops on them. A scarecrow, when placed on a spot, covers the cell to its immediate left and right along with the cell it is on.



Given the description of the field, what is the minimum number of scarecrows that needs to be placed so that all the useful section of the field is covered? Useful section refers to cells where crops can be grown.

Input

Input starts with an integer T (≤ 50), denoting the number of test cases.

Each case starts with a line containing an integer N ($0 < N < 100$). The next line contains N characters that describe the field. A dot (.) indicates a crop-growing spot and a hash (#) indicates an infertile region.

Output

For each case, print the case number and the number of scarecrows that need to be placed.

Sample Input	Output for Sample Input
3 3 .#. . 11##....## 2 ##	Case 1: 1 Case 2: 3 Case 3: 0

1390 - Weight Comparison

Rimi has n different objects having distinct weights and identified by x_1, x_2, \dots, x_n . All she wanted to do is to sort them in ascending order according to their weights. That's why she asked her robot-helper to do all the weight comparisons. She went for another task, and the robot continued doing the comparisons. This robot was not that intelligent, that's why it was picking two objects randomly, compared them and wrote down the object names as $x_i x_j$ meaning that x_i is heavier than x_j . After a while, Rimi came back and found this foolishness of the robot. So, she wants to find the comparisons that are really necessary. For example, the robot compared and found the following:

1. $x_1 > x_2$
2. $x_2 > x_5$
3. $x_1 > x_5$

Clearly the third one is unnecessary, as from the first two relations, it's clear that x_1 is heavier than x_5 . But the second one is necessary since from first and third relation, it's impossible to determine the relation between x_2 and x_5 . The first relation is also necessary. Now Rimi wants to keep minimum number of necessary relations such that they are enough to find all the comparisons.

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers n, m ($1 \leq n \leq 5000, 0 \leq m \leq 10^5$), where m denotes the number of comparisons done by the robot. Each of the next m lines contains two integers $i j$ ($1 \leq i, j \leq n, i \neq j$) meaning that x_i is heavier than x_j . There is no duplicate transition. And assume that the data is valid.

Output

For each case, print the case number and the number of necessary comparisons. Then print the comparisons in $i j$ (x_i is heavier than x_j) form, first sorted by i then by j in ascending order.

Sample Input	Output for Sample Input
2 4 4 1 2 3 4 2 3 1 4 4 1 4 2	Case 1: 3 1 2 2 3 3 4 Case 2: 1 4 2

Note

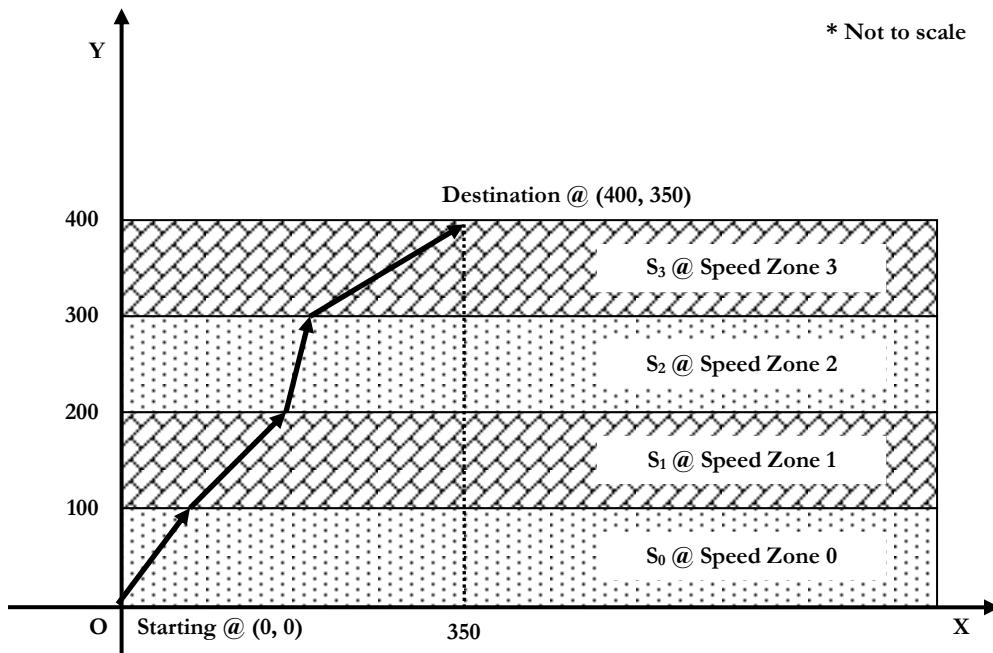
Dataset is huge; use faster I/O methods.

1391 – Speed Zones

Suppose you are in a **2-Dimensional** world. Now, you are in a system of '**N**' parallel zones of **same** or **different speed**, numbered from **0** to **N-1**. In each zone you can move in some given constant speed (S_i amount per second in i^{th} zone) at any direction. Each zone is parallel to **X** axis, starting from the **X** axis (and then on the positive **X** and positive **Y** part only). Width of each zone is **100** (along the **Y** axis).

You are currently in the origin **(0, 0)**. You need to reach **(100*N, D)** coordinate. But, you want to do that in minimum possible time (seconds).

Here is an example with **N = 4**, and **D = 350**. The arrows show **a possible path** from **(0, 0)** to **(400, 350)**. Note that after the end of each zone (except the last one), it is possible that you may be in an **non-integer 'X'** coordinate.



Given **N**, **D**, and the speeds $S_0, S_1, S_2, \dots, S_{N-1}$ you will need to find the minimum possible time in seconds to reach the destination point.

Input

Input starts with an integer **T** (≤ 50), denoting the number of test cases.

Each case contains two lines. In the **first** line you will be given two integers **N** ($1 \leq N \leq 100$) and **D** ($0 \leq D \leq 10000$). In the **second** line you will be given **N** integers, the speeds, in the order: $S_0, S_1, S_2, \dots, S_{N-1}$. You can assume that $1 \leq S_i \leq 1000$ for all $0 \leq i < N$.

Output

For each case, print the case number and the minimum possible time in seconds. Error less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
2 1 0 50 3 400 10 10 10	Case 1: 2 Case 2: 50.0000000

1392 - Multi Round Matches

"Wicked" is a popular game in country Moderdesh but the national Wicked team is losing consistently and comprehensively to other visiting foreign teams. So, MWB (Moderdesh Wicked Board) is facing a lot of trouble as local people do not turn out to see matches with international teams. So they are planning to adopt a new policy that will bring a lot of people to the stadium.

Wicked is a lengthy game that is played between robots built by two countries. So there are actually two teams of robots (it is said that all the human players got injured in an incident when angry mob attacked busses full of players in 2010 BC). The game can have as many as 10^8 rounds and in each round a team can score points within **1** to **N** (inclusive).

One of the teams plays first and scores **T** points in **R** rounds. Then the other team plays and try to score exactly **T** points in exactly **R** rounds. If the team playing second scores more than **T** points in **R** rounds then the game is a draw, if they score less than **T** points then the team playing first wins. If the team playing second scores exactly **T** points in **R** rounds, it wins the game. But MWB has changed the rules a bit to shorten the game. So first they allow a team to play exactly **R** rounds and suppose they score **T** points. But they don't allow the 2nd team to play the full **R** rounds (as in that case it is almost likely that the local team will lose), they rather give the 2nd team an intermediate state to start with. An intermediate state means that that they declare that the 2nd team has scored T_{int} points in R_{int} ($1 \leq T_{int} \leq T$, $1 \leq R_{int} \leq R$) rounds, now they have to score exactly $(T - T_{int})$ points in exactly $(R - R_{int})$ rounds to win. This situation may be very easy or very difficult for the 2nd team so this gives the local team often some very good chance to win. But it is not possible to win from all intermediate states (as one can earn minimum **1** and maximum **N** points from each round). The states from which the 2nd team can manage a win are called winning states. Given the value of **R**, **T** and **N** your job is to find out the total number of winning states.

Input

Input starts with an integer **T** (≤ 1000), denoting the number of test cases.

Each case starts with a line containing three integers **R** ($10 \leq R \leq 10^8$), **T** ($10 \leq T \leq 4*10^8$) and **N** ($2 \leq N \leq 20$). The meaning of **R**, **T** and **N** is given in the problem statement.

Output

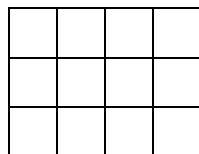
For each case, print the case number and the total number of winning states.

Sample Input	Output for Sample Input
2 10 20 5 10 20 3	Case 1: 83 Case 2: 59

1393 – Crazy Calendar

2011 was a crazy year. Many people all over the world proposed on 11-11-11, married on 11-11-11, some even went through surgery only to have 11-11-11 as their child's birth date. How crazy people can be! Don't they see there is a "20" hidden? Then what to do? A very elegant solution came from ARR, a very famous and funny character - why do we need to follow Christian (or some calls it Gregorian) calendar? Why don't we start our own calendar on the day of marriage? And those who like to celebrate their marriage ceremony too frequent, why don't they declare only 1 day per year. In that fashion they can celebrate their anniversary every day. And may be one minute a year or a second or ... Uh.. getting complex. Let's back to the title. From now, we start to have a new calendar system, "Kisu Pari Na". And we hope to update this calendar on every national contest.

The purpose of this calendar is - we all will try our best to learn something new in every year. For this first year let's learn some combinatory. It reminds me of my first year in college. I faced this problem but could not solve this then. But see how easy it is:



Say you start from upper left cell and want to go to lower right cell. The only restriction is you can only move downward or rightward. How many ways are there? How to solve it? Not that difficult. You have to go two times Down and three times Right (whichever way you try) to reach the goal from the starting cell, right? So the answer is number of ways you can arrange two **D** (represents Down) and three **R** (represent Right). 2 same characters and 3 same characters, total 5 characters. So it is:

$$\frac{5!}{2! 3!} \text{ Or } \frac{(D+R)!}{D! R!} = D+R C_R. \text{ Easy isn't it?}$$

Ok enough with learning. Now back to problem, given a grid and at each cell there are some coins. Inky and Pinky are playing a game getting inspiration from the above problem. At each turn, a player chooses a non empty cell and then removes one or more coins from that cell and put them to the cell exactly right of it or exactly beneath it. A player can't divide the coins and put one part to right and others to down. Note that, for the cells at the right column the player can't move it to more right, and same for the bottom-most row. So a player can't move coins from the lower right cell. The game will finish when no moves are available and the player who moved last will win. Now inky being very modest asked Pinky to move first. Can you say if Pinky will win if both play perfectly?

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers **R C ($1 \leq R * C \leq 50000$)**, where **R** denotes the number of rows and **C** denotes the number of columns of the grid respectively. Each of the next **R** lines contains **C** space separated integers denoting the grid. These integers lie in the range **[0, 10^9]**.

Output

For every test case, output case number followed by "**win**" if Pinky can win or "**lose**".

Sample Input	Output for Sample Input
1 2 2 1 1 1 1	Case 1: lose

Note

Dataset is huge, use faster I/O methods.

1394 - Disable the Wand

The battle of Hogwarts is going to start very soon. Hermione has reviewed the whole strategy of the battle and she finds that they need to disable some wands of some death eaters. It is very difficult to perform magic without wands, so they will have some significant advantages over those death eaters. She quickly discovers that to disable a wand they have to know the sum of the magical numbers corresponding to that wand.

The properties of a magical number of a wand are given below:

1. The number must be greater than or equal to a certain number **start**.
2. The number must be less than or equal to a certain number **end**.
3. The binary representation of a magical number contains at most **Maxone** number of ones.
4. The binary representation of a magical number can differ from **Ideal Number** at not more than **k** positions. For example, 0110_2 (6_{10}) differs from 1010_2 (10_{10}) at two positions.
5. A magical number must be an integer which is divisible by 3, but not divisible by 7.

As both start and end can be quite large, she needs your help to find the sum of the magical numbers within this range.

Input

Input starts with an integer **T** (≤ 130), denoting the number of test cases.

For each case, a single line follows which contains five integers: **start**, **end**, **Maxone**, **Ideal Number**, **k**, respectively. All of them are non-negative integers and less than or equal to 10^9 . And **start** will not be greater than **end**.

Output

For each case, print the case number and sum of the magical numbers.

Sample Input	Output for Sample Input
2 1 6 2 3 1 1 6 2 3 2	Case 1: 3 Case 2: 9

Note

For computing the number of positions of difference from the ideal number, both ideal number and magical number can be considered as 32 bit binary numbers with necessary number of leading zeros.

1395 – A Dangerous Maze (II)

You are in a maze; seeing **n** doors in front of you in beginning. You can choose any door you like. The probability for choosing a door is equal for all doors.

If you choose the **ith** door, it can either take you back to the same position where you begun in **x_i** minutes, or can take you out of the maze after **x_i** minutes. If you come back to the same position, you can remember last **K** doors you have chosen. And when you are about to choose a door, you never choose a door that is already visited by you. Or we can say that you never choose a door that is visited as one of the last **K** doors. And the probability of choosing any remaining door is equal.

Now you want to find the expected time to get out of the maze.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains a blank line and two integers **n K** ($1 \leq n \leq 100$, $0 \leq K \leq n$). The next line contains **n** space separated integers. If the **ith** integer (**x_i**) is positive, you can assume that the **ith** door will take you out of maze after **x_i** minutes. If it's negative, then the **ith** door will take you back to the beginning position after **abs(x_i)** minutes. You can safely assume that $1 \leq \text{abs}(x_i) \leq 10000$.

Output

For each case, print the case number and the expected time to get out of the maze. If it's impossible to get out of the maze, print '**-1**'. Otherwise print the result. Error less than 10^{-6} will be ignored.

Sample Input	Output for Sample Input
4 2 0 10 10 2 0 10 -10 3 1 10 -10 -20 3 2 10 -10 -20	Case 1: 10 Case 2: 20.000 Case 3: 30.0000000000 Case 4: 25.0000000000

1396 – Palindromic Numbers (III)

Vinci is a little boy and is very creative. One day his teacher asked him to write all the Palindromic numbers from 1 to 1000. He became very frustrated because there is nothing creative in the task. Observing his expression, the teacher replied, "All right then, you want hard stuff, you got it." Then he asks Vinci to write a palindromic number which is greater than the given number. A number is called palindromic when its digits are same from both sides. For example: 1223221, 121, 232 are palindromic numbers but 122, 211, 332 are not. As there can be multiple solutions, Vinci has to find the number which is as small as possible.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing a positive integer. This integer can be huge and can contain up to **10^5** digits.

Output

For each case, print the case number and the minimum possible palindromic number which is **greater** than the given number.

Sample Input	Output for Sample Input
5 121 1 1332331 11 1121	Case 1: 131 Case 2: 2 Case 3: 1333331 Case 4: 22 Case 5: 1221

Note

Dataset is huge, use faster I/O methods.

1397 - Sudoku Solver

Sudoku is a logic-based, combinatorial, number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid (also called "boxes", "blocks", "regions", or "sub-squares") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which typically has a unique solution. Or we can say, the same single integer may not appear twice in the same 9×9 playing board row or column or in any of the nine 3×3 sub-regions of the 9×9 playing board. Now you are given a partially filled Sudoku board that has a unique solution. Your task is to fill the board.

	Given Sudoku Puzzle		Puzzle with Solution
--	---------------------	--	----------------------

Input

Input starts with an integer T (≤ 30), denoting the number of test cases.

Each case starts with a blank line. Then there will be 9 lines, each containing 9 characters denoting the board as described. Empty places will be marked by a '!'.

Output

For each case, print the case number in a single line. Then print the solution in 9 lines with 9 characters in each line.

Sample Input	Output for Sample Input
1 .46....9.. .3.1..... .2..6..85 ...87..... 6...3....414... 79...5..3.2.4. ..2...61.	Case 1: 146528973 835197462 927463185 459876321 618235794 273914856 794651238 361782549 582349617

1398 - Winger Trial

After the great winger Donaldo left his soccer team, coach sir Thelex has found himself in a great fix. The strength of his team is reduced greatly and he needs to find a suitable replacement immediately. The coach selects a number of young wingers from around the world and sets up a trial for them.

The trial will take place on a rectangular shaped field of length **L** meters and width **W** meters. There are **N** robot defenders placed on the field. The defenders do not change their positions but if a winger's distance from a defender is not more than **d** meters, it will automatically tackle him. A robot defender may tackle at most once. On the beginning of the trial, a winger stands on the left edge of the field (across the length) with a soccer ball. Now, his task is to avoid the obstructions of the robot defenders and reach the rightmost edge of the field with the ball. Please tell him the minimum number of tackles he must face in order to reach the opposite end. A player must not go outside the field or he will be disqualified.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing four integers **L** **W** **N** **d** ($1 \leq L, W \leq 10000, 1 \leq N \leq 100, 1 \leq d \leq 1000$) as described above. Each of the following **N** lines contains two integers defining the **x** and **y** co-ordinates of a defender. You can consider the co-ordinate of the lower-left corner of the field to be **(0, 0)** and upper-right corner to be **(L, W)**. Obviously, all the defenders are located inside the field. The left edge is denoted by the line **(0, 0), (0, W)** and the right edge is denoted by the line **(L, 0), (L, W)**. The winger wants to move from left edge to right.

Output

For each case, print the case number and the minimum number of tackles that needs to be dealt with.

Sample Input	Output for Sample Input
2 500 300 5 100 250 0 250 150 250 300 100 150 400 150 10 10 4 4 2 3 8 3 2 7 8 7	Case 1: 1 Case 2: 2

1399 – Politeness

A number is called polite if it can be written as a sum of some (at least two) consecutive positive integers. For example, 6 ($1+2+3$) is a polite number while 4 is not. Politeness of a number is the number of ways it can be expressed as sum of consecutive positive integers. For example 6($1+2+3$) is a politeness of 1 while $18(5+6+7 = 3+4+5+6)$ has a politeness of 2. Obviously, non polite number has a politeness of 0.

You are given a politeness **P**, you have to find out the smallest number that has the politeness **P**.

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer **P** ($0 \leq P \leq 10^{12}$).

Output

For each case, print the case number and the integer that has the politeness **P**. If there are several integers having politeness **P**, choose the smallest one. As the integer can be big; print the integer modulo **1000 000 007**.

Sample Input	Output for Sample Input
7	Case 1: 1
0	Case 2: 3
1	Case 3: 45
5	Case 4: 343299432
852281891999	Case 5: 129653419
975018442254	Case 6: 221997673
986350945007	Case 7: 3917908
511	

Note

For case 7, 1003917915 is the smallest number with politeness 511, so the result is (1003917915 modulo 1000000007), which is 3917908.

1400 - Employment

Employment is a contract between two parties, one being the employer and the other being the employee. Now assume that there are n companies who require **one** employee each and there are n candidates. All the candidates interviewed in each of the companies and eventually they have different preferences over the companies and the companies have different preferences over the candidates.

So, you are given the task to assign each candidate to each company such that the employment scheme is stable. A scheme is stable if there is no pair (**candidate_i, company_j**) and (**candidate_x, company_y**) where

- a) **Candidate_i** prefers **company_y** more than **company_j** and
- b) **Company_y** prefers **candidate_i** more than **candidate_x**.

As there can be many solutions, any valid one will do.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing an integer **n ($1 \leq n \leq 100$)**. The candidates are numbered from **1** to **n** and the companies are numbered from **n+1** to **2n**.

Each of the next **n** lines contains **n** distinct integers from **n+1** to **2n**, where the **ith** line contains the company preference for the **ith** candidate (**1 ≤ i ≤ n**).

Each of the next **n** lines contains **n** distinct integers from **1** to **n**, where the **ith** line contains the candidate preference for the company which is denoted by **n+i** (**1 ≤ i ≤ n**).

Output

For each case, print the case number and the (candidate, company) pairs. As there can be many solutions any valid one will do. And you can output the pairs in any order but print those as (candidate, company) pair.

Sample Input	Output for Sample Input
1 3 4 5 6 6 5 4 5 4 6 2 1 3 1 2 3 3 2 1	Case 1: (2 6) (1 4) (3 5)

Note

This is a special judge problem; wrong output format may cause 'Wrong Answer'.

1401 - No More Tic-tac-toe

Alice was bored with the game tic-tac-toe. Usually she used to play against computer. She was so bored because, she always managed to make a tie against computer as she already became an expert in this game. She was also bored about the matching things, that means she does not like to see two adjacent '**X**'s or two adjacent '**O**'s. So Bob created a new computer game for her.

Here is the description of this new 2-player game:

- 1) A **1 x N** grid will be given.
- 2) In each turn, a player can mark an unmarked cell by an '**X**' or by an '**O**' (it's not Zero it is big '**O**').
- 3) No two adjacent '**X**'s are allowed in this game.
- 4) No two adjacent '**O**'s are allowed in this game.
- 5) The player who marks the last cell wins the game.

Let's have an example for **N = 3**. Suppose first player mark the leftmost cell with an '**X**'. Then the grid will look like this:



From this state, the winning move of the second player is to mark the rightmost cell with an '**O**'.



Now first player has no move, because in the only empty cell she cannot mark '**X**' and cannot mark '**O**' because the left of this cell is already marked '**X**' and the right of this cell is marked '**O**'. The second player will win.

But this is not an optimal example for first player. If in first move, the first player marks the middle cell with any one of '**X**' or '**O**', she will win.

Now about Bob's computer game, Alice always makes the first move. Alice usually used to start playing with Bob. But most often after some move, Bob gave his responsibility to the computer. Computer always plays optimally. You are given the state of the grid after Bob's departure and your task is to determine whether it is possible to win against computer from this state for Alice.

Input

Input starts with an integer **T (≤ 10000)**, denoting the number of test cases.

Each case starts with a line containing a string which represents the state of the game after Bob's departure. Here the length of the string is **N ($1 \leq N \leq 100$)**. In this string besides '**X**' and '**O**', another character '**.**' is used, which represents an unmarked cell.

Output

For each case, print the case number and 'Yes' if it is possible to win against computer, otherwise print 'No'.

Sample Input	Output for Sample Input
5 ... X..O ..X.. O	Case 1: Yes Case 2: No Case 3: No Case 4: Yes Case 5: Yes

1402 – Line Chart

WARush is very famous in super-coder. In super-coder algorithm contest rank-list WARush is ranked one. Now-a-days, he is doing some analysis on his rating history in super-coder algorithm contest. In super-coder an algorithm contest is said as a Single Round Tournament (SRT). After each SRT finished, rating of a contestant is updated according to his/her performance. WARush collected all this rating information, and using this he created a line chart.

To make things more clear, let us consider the following table as his rating info.

SRT	Rating
320	3
306	1
401	3
325	4
393	5
380	2

From this table, we see that his first SRT was SRT-306, and rating after that SRT was 1, so he marked point $(1, 1)$ as r_1 in graph paper, his second SRT was SRT-320 and rating after that SRT was 3, so he marked $(2, 3)$ as r_2 , then he add r_1 with r_2 by a straight line and so on.

In general for his i^{th} SRT he marked point $(i, \text{rating after } i^{\text{th}} \text{ SRT})$ by r_i . After marking all the points he will add point r_i with r_{i-1} by straight lines, for all $1 < i \leq N$, Where N is the number of SRTs he participated. Look at figure 1 for more clear idea.

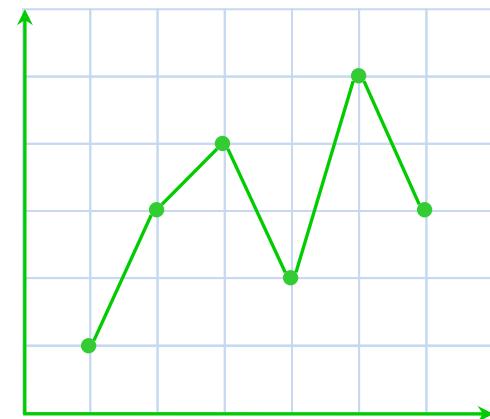


Fig 1: Line Chart Considering all SRTs

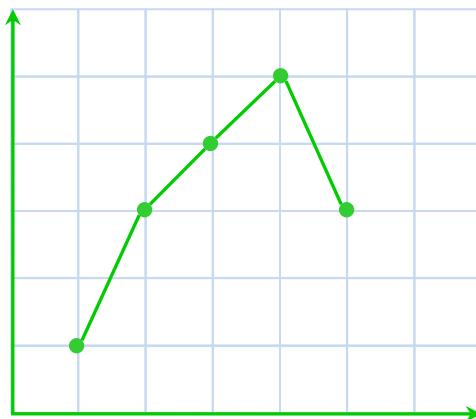


Fig 2: Line Chart Ignoring SRT 380

After drawing line chart, he became very interested about the number of peaks. There are two kinds of peaks in a line chart, 1) Upper Peak and 2) Lower Peak. **Upper Peak** is those points in a line chart, both of previous and next point of which have smaller y coordinates, and **Lower Peak** is those points in a line chart, both of previous and next point of which have greater y coordinates.

For example total number of peak in figure 1 is 3. Two of them are upper peaks, which are (3, 4) and (5, 5), and one of them is lower peak which is (4, 2).

WARush observed that by ignoring SRT-380, his line chart will become same as figure 2, in which number of peaks is only 1. By observing this he became more curious. Now he wants to know, by ignoring 0 or more SRTs, how many distinct line chart having **K** peaks can be possible. WARush called these line charts "**K-peak Line charts**", in a **K-peak** line chart he doesn't allow two consecutive points having same y coordinate.

Input

Input starts with an integer **T** (≤ 12), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **N** ($1 \leq N \leq 10000$) and **K** ($0 \leq K \leq 50$). Each of the next **N** lines contains two integers, **SRT_i** ($1 \leq SRT_i \leq 10^9$) and **Rating_i** ($1 \leq Rating_i \leq 10^9$). You can safely assume that all the SRTs are distinct.

Output

For each case, print the case number and the number of distinct **K-peak** line charts modulo 2^{32} .

Sample Input	Output for Sample Input
3 6 1 320 3 306 1 401 3 325 4 393 5 380 2 4 1 101 3 102 2 103 2 104 4 3 0 102 2 101 1 103 3	Case 1: 20 Case 2: 1 Case 3: 8

1403 - Air Raid

Consider a town where all the streets are one-way and each street leads from one intersection to another. It is also known that starting from an intersection and walking through town's streets you can never reach the same intersection i.e. the town's streets form no cycles. With these assumptions your task is to write a program that finds the minimum number of paratroopers that can descend on the town and visit all the intersections of this town in such a way that no intersection is visited by more than one paratrooper. Each paratrooper lands at an intersection and can visit other intersections following the town streets. There are no restrictions on the starting intersection for each paratrooper.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n ($1 \leq n \leq 1000$)** and **m ($0 \leq m \leq 10000$)** where **n** denotes the number of intersections and **m** denotes the number of one-way streets. Intersections are numbered from **1** to **n**. Each of the next **m** lines contains two integers **u v ($1 \leq u, v \leq n, u \neq v$)** denoting a one-way street from intersection **u** to **v**. There can be at most one street from one intersection to another.

Output

For each case, print the case number and the minimum number of paratroopers required to visit all the intersections in the town.

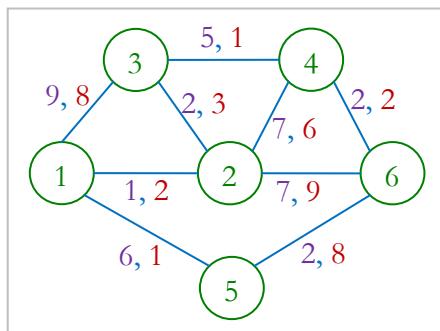
Sample Input	Output for Sample Input
3 4 3 3 4 1 3 2 3	Case 1: 2 Case 2: 1 Case 3: 3
3 3 1 3 1 2 2 3	
5 4 1 2 3 2 2 4 2 5	

Note

Dataset is huge, use faster I/O methods.

1404 - Sending Secret Messages

Alice wants to send Bob some confidential messages. But their internet connection is not secured enough. As their names have been used in many networking schemes, they are very rich now. So, they don't want to send encoded messages, they want to use secured dedicated connection for them. So, they talked to some ISPS (Internet Service Providers) about their problem. Only they get is that there are **N** routers in the network, some of them share bidirectional links. Each link has a capacity, and for each KB of data passing through this link, they have to pay some money. Assume that Alice is connected with the **1st** router and Bob is connected to the **Nth** router.



For example, in the picture, Alice wants to send 4 KB data from router 1 to router 6. Each link is identified by two integers in the form **(a, b)** where '**a**' denotes the capacity of the link and '**b**' denotes per KB cost of the link. So, Alice can send 1KB of data through 1 - 2 - 3 - 4 - 6 (cost 8), 2KB data through 1 - 5 - 6 (cost $2 * 9 = 18$) and 1KB data through 1 - 3 - 4 - 6 (cost 11). So, the total cost is 37 units.

Now Alice wants to send **P** KB of data to Bob. You have to find the minimum amount of money they have to pay to achieve their goal.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers **N ($2 \leq N \leq 50$)**, **M ($0 \leq M \leq N*(N-1)/2$)** and **P ($1 \leq P \leq 1000$)**, where **M** denotes the number of bidirectional links. Each of the next **M** lines contains four integers **u v w c ($1 \leq u, v \leq N, u \neq v, 1 \leq w, c \leq 100$)**, meaning that there is a link between router **u** and **v**, and at most **c** **KB** data can be sent through this link, and each KB of data through this link will cost **w**. You can assume that there will be at most one connection between a pair of routers.

Output

For each case, print the case number and the minimum amount of money required or "**impossible**" if it's not possible to send **P** KB of data.

Sample Input	Output for Sample Input
<p>3</p> <p>6 9 4 3 1 9 8 1 2 1 2 1 5 6 1 5 6 2 8 6 4 2 2 4 2 7 6 2 6 7 9 3 4 5 1 3 2 2 3</p> <p>6 9 9 3 1 9 8 1 2 1 2 1 5 6 1 5 6 2 8 6 4 2 2 4 2 7 6 2 6 7 9 3 4 5 1 3 2 2 3</p> <p>4 4 20 1 3 1 3 3 4 1 4 1 2 1 2 2 4 1 5</p>	<p>Case 1: 37 Case 2: 139 Case 3: impossible</p>

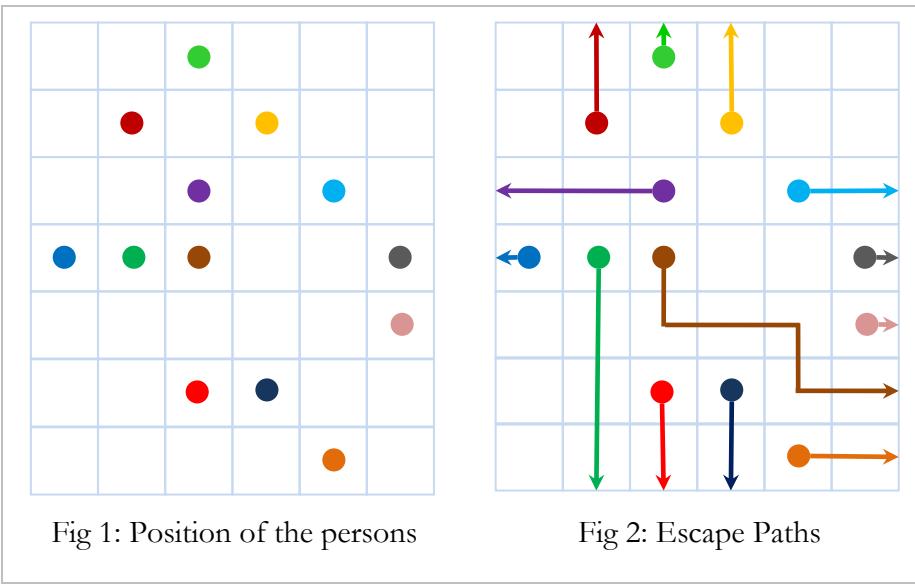
1405 – The Great Escape

Walter White finally discovered the medicine for a dangerous disease. That's why he is in danger, because some people are targeting him to get the formula. So, now he and his co-workers have to leave the city.

Assume that the city can be modeled as an $M \times N$ grid, where a cell containing a '*' means that a co-worker lives in that place, a cell containing a '!' means it's empty.

So, Walter informed all of them by phone to leave the city at once and he has laid down a rule that no one will cross path in their way out of the city. Since if two or more persons meet at same cell (even in the boundary cells), there is a big possibility that they all might get caught. Walter doesn't want that, so he needs to know whether all of them can get out of the city. So, he asked you to help him.

For simplicity, assume that if one person reaches any of the border cells of the grid, he is considered to be out of the city. And from any cell, a person can move only to its four adjacent cells (north, south, east or west).



For example, for the given 7×6 grid, the colored circles show the position of the persons who are escaping (fig 1). And a feasible solution is shown in fig 2.

Input

Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case starts with a line containing two integers M and N ($1 \leq M, N \leq 100$). Each of the next M lines contains N characters denoting the map. Each character is either '!' or '*'. Total number of persons in a map will be at most $2*(M+N)$.

Output

For each case, print the case number and 'yes' if it's possible for them to get out of the city maintaining the restrictions. Otherwise print 'no'.

Sample Input	Output for Sample Input
3 7 6 . * * . * * . * . *** . . * * . . * * * . 4 10 *** .. *** . * * . ** . *** . * . . * . . * . . *** . ** . * . * . ***	Case 1: yes Case 2: no Case 3: yes
4 10 *** .. *** . * * . ** . *** . * . . * . . * . . . * . ** . * . * . ***	

1406 - Assassin's Creed



Altair is in great danger as he broke the three tenets of the assassin creed. The three tenets are: 1) never kill an innocent people, 2) always be discrete and 3) never compromise the brotherhood. As a result Altair is given another chance to prove that he is still a true assassin. Altair has to kill **n** targets located in **n** different cities. Now as time is short, Altair can send a message along with the map to the assassin's bureau to send some assassins who will start visiting cities and killing the targets. An assassin can start from any city, but he cannot visit a city which is already visited by any other assassin except him (because they do not like each other's work). He can visit a city multiple times though. Now Altair wants to find the minimum number of assassins needed to kill all the targets. That's why he is seeking your help.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n ($1 \leq n \leq 15$)** and **m ($0 \leq m \leq 50$)**, where **n** denotes the number of cities and **m** denotes the number of one way roads. Each of the next **m** lines contains two integers **u v ($1 \leq u, v \leq n, u \neq v$)** meaning that there is a road from **u** to **v**. Assume that there can be at most one road from a city **u** to **v**.

Output

For each case, print the case number and the minimum number of assassins needed to kill all the targets.

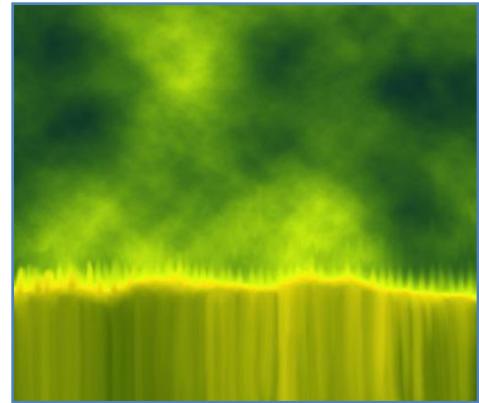
Sample Input	Output for Sample Input
2 3 2 1 2 2 3 6 6 1 2 2 3 2 4 5 4 4 6 4 2	Case 1: 1 Case 2: 2

1407 - Explosion

Planet Krypton is about to explode. The inhabitants of this planet have to leave the planet immediately. But the problem is that, still some decisions have to be made - where to go, how to go etc. So, the council of Krypton has invited some of the people to meet in a large hall.

There are **n** people in planet Krypton, for simplicity they are given ids from **1** to **n**. The council uses a super computer named Oracle to call them in the meeting. Oracle has four types of messages for invitation. The message format is **type** **x** **y**, where **x** and **y** are two different person's ids and **type** is an integer as follows:

1. **1** **x** **y** means that either **x** or **y** should be present in the meeting.
2. **2** **x** **y** means that if **x** is present, then no condition on **y**, but if **x** is absent **y** should be absent
3. **3** **x** **y** means that either **x** or **y** must be absent.
4. **4** **x** **y** means that either **x** or **y** must be present but not both.



Each member of the council has an opinion too. The message format is **type** **x** **y** **z**, where **x**, **y** and **z** are three different person's ids and **type** is an integer as follows:

1. **1** **x** **y** **z** means that at least one of **x**, **y** or **z** should be present
2. **2** **x** **y** **z** means that at least one of **x**, **y** or **z** should be absent

Now you have to find whether the members can be invited such that every message by oracle and the council members are satisfied.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers **n**, **m** and **k** ($3 \leq n \leq 1000$, $0 \leq m \leq 2000$, $0 \leq k \leq 5$) where **m** means the number of messages by oracle, **k** means the total members in the council. Each of the next **m** lines will contain a message of Oracle in the format given above. Each of the next **k** lines will contain a message of a council member. You can assume that all the ids given are correct.

Output

For each case, print the case number and whether it's possible to invite the people such that all the messages are satisfied. If it's not possible, then print '**Impossible.**' in a single line. Otherwise, print '**Possible**' and the number of invited people and the ids of the invited people in ascending order. Print the line leaving a single space between fields. Terminate this line with a '!'. See the samples for more details. There can be multiple answers; print any valid one.

Sample Input	Output for Sample Input
<pre> 3 3 2 1 3 2 1 1 2 3 1 1 2 3 4 4 1 2 2 1 4 1 2 4 1 3 4 1 4 2 2 3 4 4 5 0 3 1 2 2 2 3 2 2 4 2 1 2 2 2 1 </pre>	<pre> Case 1: Possible 2 1 3. Case 2: Impossible. Case 3: Possible 0. </pre>

Note

This is a special judge problem; wrong output format may cause 'Wrong Answer'.

1408 - Batting Practice

After being all out for 58 and 78 in two matches in the most prestigious tournament in the world, the coach of a certain national cricket team was very upset. He decided to make the batsmen practice a lot. But he was wondering how to make them practice, because the possibility of getting out seems completely random for them. So, he decided to keep them in practice as long as he can and told them to practice in the net until a batsman remains not-out for k_1 consecutive balls. But if the batsman continues to be out for consecutive k_2 balls, then the coach becomes hopeless about the batsman and throws him out of the team. In both cases, the practice session ends for the batsman. Now the coach is wondering how many balls the practice session is expected to take.



For a batsman the probability of being out in a ball is independent and is equal to p . What is the expected number of balls he must face to remain not out for k_1 consecutive balls or become out in consecutive k_2 balls.

Input

Input starts with an integer T (≤ 15000), denoting the number of test cases.

Each case starts with a line containing a real number p ($0 \leq p \leq 1$) and two positive integers k_1 and k_2 ($k_1 + k_2 \leq 50$). p will contain up to three digits after the decimal point.

Output

For each case, print the case number and the expected number of balls the batsman will face. Errors less than 10^{-2} will be ignored.

Sample Input	Output for Sample Input
5 0.5 1 1 0.5 1 2 0.5 2 2 0.19 1 3 0.33 2 1	Case 1: 1 Case 2: 1.5 Case 3: 3 Case 4: 1.2261000000 Case 5: 1.67

1409 – Rent a Car

The **ACM (Advanced Car Management) Rent a Car company** is very famous now-a-days because of their quality and service. Gaining popularity is not that easy as there are many competitor companies around. Each day they have a large number of car requests. Once a car is used for a day, if they want to use it later, they should send it for servicing. Actually it was their key theme for business and that's why they are so popular.

There are C motor companies in town, where the k^{th} company has c_k cars in their showroom and price of a car of this company is p_k . There are R car service-centers in town, the i^{th} center takes d_i days and costs s_i per car service. Service centers can service huge number of cars at the same time.

Now, ACM company has the request sheet for next N days, where in j^{th} day, r_j cars are needed. They want to fulfill all the requirements with minimized cost. Initially, ACM has empty garage. But their garage is huge and can store any number of cars.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with three integers N, C, R ($1 \leq N, C, R \leq 50$). The next line contains N integers where the j^{th} integer denotes r_j ($0 \leq r_j \leq 100$). The next line contains $2C$ integers where the k^{th} integer-pair denotes c_k and p_k ($1 \leq c_k, p_k \leq 100$). The next line contains $2R$ integers where the i^{th} integer-pair denotes d_i and s_i ($1 \leq d_i, s_i \leq 100$).

Output

For each case, print the case number and the minimized cost to fulfill all the requests. If it's impossible to do so, print "**impossible**".

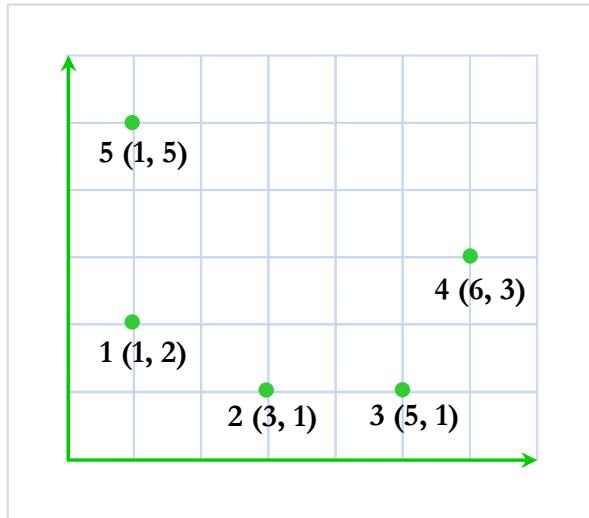
Sample Input	Output for Sample Input
2 3 2 1 10 20 30 40 90 15 100 1 5 3 2 1 10 20 30 40 90 15 100 2 5	Case 1: 4650 Case 2: impossible

Note

For case 1, 50 cars will be bought, 40 from company 1 (costs $40*90=3600$) and 10 from company 2 (costs $10*100=1000$). On day 1, 10 cars will be sent and then they will be sent to the service center (costs $10*5=50$). The cars will be received on day 3. On day 2, 20 cars will be sent. And on day 3, 20 unused cars will be sent along with the 10 cars (serviced). So, overall cost is $3600+1000+50 = 4650$.

1410 - Consistent Verdicts

In a 2D plane N persons are standing and each of them has a gun in his hand. The plane is so big that the persons can be considered as points and their locations are given as Cartesian coordinates. Each of the N persons fire the gun in his hand exactly once and no two of them fire at the same or similar time (the sound of two gun shots are never heard at the same time by anyone so no sound is missed due to concurrency). The hearing ability of all these persons is exactly same. That means if one person can hear a sound at distance R_1 , so can every other person and if one person cannot hear a sound at distance R_2 the other $N-1$ persons cannot hear a sound at distance R_2 as well.



The N persons are numbered from 1 to N . After all the guns are fired, all of them are asked how many gun shots they have heard (not including their own shot) and they give their verdict. It is not possible for you to determine whether their verdicts are true but it is possible for you to judge if their verdicts are consistent. For example, look at the figure above. There are five persons and their coordinates are $(1, 2)$, $(3, 1)$, $(5, 1)$, $(6, 3)$ and $(1, 5)$ and they are numbered as 1, 2, 3, 4 and 5 respectively. After all five of them have shot their guns, you ask them how many shots each of them have heard. Now if their response is 1, 1, 1, 2 and 1 respectively then you can represent it as $(1, 1, 1, 2, 1)$. But this is an inconsistent verdict because if person 4 hears 2 shots then he must have heard the shot fired by person 2, then obviously person 2 must have heard the shot fired by person 1, 3 and 4 (person 1 and 3 are nearer to person 2 than person 4). But their opinions show that Person 2 says that he has heard only 1 shot. On the other hand $(1, 2, 2, 1, 0)$ is a consistent verdict for this scenario so is $(2, 2, 2, 1, 1)$. In this scenario $(5, 5, 5, 4, 4)$ is not a consistent verdict because a person can hear at most 4 shots.

Given the locations of N persons, your job is to find the total number of different consistent verdicts for that scenario. Two verdicts are different if opinion of at least one person is different.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing a positive integer N ($1 \leq N \leq 700$). Each of the next N lines contains two integers x_i y_i ($0 \leq x_i, y_i \leq 30000$) denoting a co-ordinate of a person. Assume that all the co-ordinates are distinct.

Output

For each case, print the case number and the total number of different consistent verdicts for the given scenario.

Sample Input	Output for Sample Input
2 3 1 1 2 2 4 4 2 1 1 5 5	Case 1: 4 Case 2: 2

1411 – Rip Van Winkle's Code

Rip Van Winkle was fed up with everything except programming. One day he found a problem which required to perform three types of update operations (**A**, **B**, **C**), and one query operation **S** over an array **data[]**. Initially all elements of **data[]** are equal to 0. Though Rip Van Winkle is going to sleep for 20 years, and his code is also super slow, you need to perform the same update operations and output the result for the query operation **S** in an efficient way.

```
long long data[250001];
void A( int st, int nd ) {
    for( int i = st; i <= nd; i++ ) data[i] = data[i] + (i - st + 1);
}
void B( int st, int nd ) {
    for( int i = st; i <= nd; i++ ) data[i] = data[i] + (nd - i + 1);
}
void C( int st, int nd, int x ) {
    for( int i = st; i <= nd; i++ ) data[i] = x;
}
long long S( int st, int nd ) {
    long long res = 0;
    for( int i = st; i <= nd; i++ ) res += data[i];
    return res;
}
```

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

Each case starts with a line containing an integer **N** ($1 \leq N \leq 10^5$) denoting the number of operations. Each of the next **N** lines starts with a character ('**A**', '**B**', '**C**' or '**S**'), which indicates the type of operation. Character '**A**', '**B**' or '**S**' will be followed by two integers, **st** and **nd** in the same line. Character '**C**' is followed by three integers, **st**, **nd** and **x**. It's assumed that, $1 \leq st \leq nd \leq 250000$ and $-10^5 \leq x \leq 10^5$. The meanings of these integers are explained in Winkle's code.

Output

For each case, print the case number first. Then for each operation '**S**', print **S(st, nd)** in a line.

Sample Input	Output for Sample Input
2 5 A 1 6 B 3 5 S 1 6 C 6 10 -2 S 1 6 1 S 1 10	Case 1: 27 19 Case 2: 0

Note

Dataset is huge, use faster I/O methods.

1412 – Visiting Islands

There are **N** islands and **M** bridges. All the bridges are setup between two islands and to pass a bridge you have to give a toll of **\$1**. The bridges are built in such a way that there is not more than one path among two islands. Now, you want to visit at least **K** different islands. You may choose the starting island of your choice, but you want to visit at least **K** different islands in minimum cost (starting island is considered to be already visited).

Input

Input starts with an integer **T** (≤ 10), denoting the number of test cases.

Each case starts with two integers **N**, **M** ($1 \leq N \leq 10^5$, $0 \leq M < N$). Each of the next **M** lines contains two integers **u v** ($1 \leq u, v \leq N$, $u \neq v$) meaning that there is a bridge between island **u** and **v**. No bridge will be reported more than once.

The next line contains an integer **q** ($1 \leq q \leq 50000$) denoting the number of queries. Each of the next **q** lines contains one integer **K** ($1 \leq K \leq 10^5$).

Output

For each case, print the case number first. Then for each query, print the minimum amount of toll you need to pay to visit at least **K** different islands. If it is not possible, print "impossible".

Sample Input	Output for Sample Input
2 2 1 1 2 3 1 2 3 5 4 1 2 2 3 2 4 2 5 2 3 2	Case 1: 0 1 impossible Case 2: 2 1

Notes

1. Dataset is huge, use faster I/O methods.
2. For the first case, for **K = 1**, which ever island we start with, we visit this. So without giving any toll we can visit one island. For **K = 2**, we choose island 1 to start. So we visit island 2 using the only bridge. So it costs \$1. For **K = 3**, as there are only 2 islands in total so we cannot visit 3 islands.

1414 – February 29

It is 2012, and it's a leap year. So there is a "February 29" in this year, which is called leap day. Interesting thing is the infant who will born in this February 29, will get his/her birthday again in 2016, which is another leap year. So February 29 only exists in leap years. Does leap year comes in every 4 years? Years that are divisible by 4 are leap years, but years that are divisible by 100 are not leap years, unless they are divisible by 400 in which case they are leap years.

In this problem, you will be given two different date. You have to find the number of leap days in between them.

Input

Input starts with an integer **T (≤ 550)**, denoting the number of test cases.

Each of the test cases will have two lines. First line represents the first date and second line represents the second date. Note that, the second date will not represent a date which arrives earlier than the first date. The dates will be in this format - "**month day, year**", See sample input for exact format. You are guaranteed that dates will be valid and the year will be in between $2 * 10^3$ to $2 * 10^9$. For your convenience, the month list and the number of days per months are given below. You can assume that all the given dates will be a valid date.

Output

For each case, print the case number and the number of leap days in between two given dates (inclusive).

Sample Input	Output for Sample Input
4 January 12, 2012 March 19, 2012 August 12, 2899 August 12, 2901 August 12, 2000 August 12, 2005 February 29, 2004 February 29, 2012	Case 1: 1 Case 2: 0 Case 3: 1 Case 4: 3

Note

The names of the months are {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November" and "December"}. And the numbers of days for the months are {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30 and 31} respectively in a non-leap year. In a leap year, number of days for February is 29 days; others are same as shown in previous line.

1415 - Save the Trees

N trees were planted on a side of a long straight road. In the other side there are farms, industries etc. The market price of these trees is huge now. Some greedy people want to cut these trees down and want to be millionaires. They got the permission from the Govt. decoying that they would develop the area. You published this fact in the web and millions raised their voices against this conspiracy.

You gathered the information of the trees and found the **type** and **height** of all trees. For simplicity, you represented them as integers. You want to find the overall price of the trees. To find the price, the following method is used.

- 1) The trees are first partitioned into groups with the condition that, types of two trees will not be similar in a group. A group can only be formed using contiguous trees.
- 2) The price of a group is equal to the height of the tallest tree.
- 3) The overall price is the summation of prices of all groups.

Now you want to find the minimum possible price of the trees in this scheme and show the Govt. that even though you calculated the minimum possible price, it's actually huge.

Input

Input starts with an integer **T** (≤ 5), denoting the number of test cases.

Each case starts with an integer **N** ($1 \leq N \leq 2 \times 10^5$). Each of the next **N** lines contains two integers **type_i**, **height_i** ($1 \leq \text{type}_i, \text{height}_i \leq 2 \times 10^5$) of the i^{th} tree from left to right.

Output

For each case, print the case number and the minimum possible price of the trees according to the scheme described above.

Sample Input	Output for Sample Input
2 5 3 11 2 13 1 12 2 9 3 13 4 3 5 2 21 3 12 2 5	Case 1: 26 Case 2: 31

Note

Dataset is huge, use faster I/O methods.

1416 – Superb Sequence

There were three friends (Alice, Bob and Carol) who regularly went to expeditions and discovered new mountain peaks. They often proposed different names and it was a problem to decide which name they would choose for the newly discovered peaks. Alice and Bob both said that the name of the peak must be a super sequence of their proposed names **A** and **B**, i.e. **A** and **B** should be **subsequences** of the name of the peak. Carol said that the name of the peak must be a **subsequence** of her proposed name **C**. As they don't like long names, they want to know the number of distinct shortest names which satisfy their needs.

So, given three strings **A**, **B** and **C**, you have to find the number of distinct shortest **common super sequences** of **A** and **B** who are also a **subsequence** of **C**. Moreover, you need to find the lexicographically earliest such sequence. Two sequences are distinct if they differ in at least one position. A **subsequence** is a sequence obtained by deleting zero or more characters from a string. A **super-sequence** is a sequence obtained by inserting zero or more characters in one or more positions of the string.

For example, say, **A** = "cdfa", **B** = "dga" and **C** = "bcdcfgaga". Then there are two shortest common super sequences of **A** and **B**: "cdfga" and "cdgfa", but "cdgfa" is not a subsequence of **C**. So the only possible name for the peak is "cdfga".

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case contains three lines. First line contains a string denoting **A**, second line contains **B** and third line contains **C**. Assume that the strings are non-empty and length of **A** and **B** will not be more than **100** and length of **C** will not be more than **300**.

Output

For each case, print the case number and the number of distinct possible shortest names for the peak modulo **1000 000 007**. And second line should contain the lexicographically earliest name. If no solution is found then print "**NOT FOUND**" in second line.

Sample Input	Output for Sample Input
2 cdfa dga bcdcfgaga abc defm abcdfghm	Case 1: 1 cdfga Case 2: 0 NOT FOUND

1417 – Forwarding Emails

"... so forward this to ten other people, to prove that you believe the emperor has new clothes."

Aren't those sorts of emails annoying?

Martians get those sorts of emails too, but they have an innovative way of dealing with them. Instead of just forwarding them willy-nilly, or not at all, they each pick one other person they know to email those things to every time - exactly one, no less, no more (and never themselves). Now, the Martian clan chieftain wants to get an email to start going around, but he stubbornly only wants to send one email. Being the chieftain, he managed to find out who forwards emails to whom, and he wants to know: which Martian should he send it to maximize the number of Martians that see it?

Input

Input starts with an integer **T** (≤ 20), denoting the number of test cases.

Each case starts with a line containing an integer **N** ($2 \leq N \leq 50000$) denoting the number of Martians in the community. Each of the next **N** lines contains two integers: **u v** ($1 \leq u, v \leq N, u \neq v$) meaning that Martian **u** forwards email to Martian **v**.

Output

For each case, print the case number and an integer **m**, where **m** is the Martian that the chieftain should send the initial email to. If there is more than one correct answer, output the smallest number.

Sample Input	Output for Sample Input
3 3 1 2 2 3 3 1 4 1 2 2 1 4 3 3 2 5 1 2 2 1 5 3 3 4 4 5	Case 1: 1 Case 2: 4 Case 3: 3

1418 – Trees on My Island

I have bought an island where I want to plant trees in rows and columns. So, the trees will form a rectangular grid and each of them can be thought of having integer coordinates by taking a suitable grid point as the origin.

But, the problem is that the island itself is not rectangular. So, I have identified a simple polygonal area inside the island with vertices on the grid points and have decided to plant trees on grid points lying strictly inside the polygon.

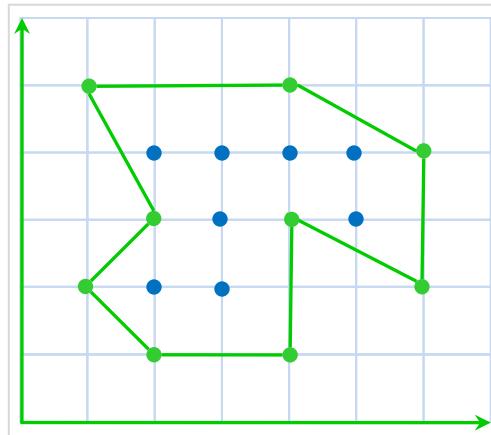


Figure: A sample of my island

For example, in the above figure, the green circles form the polygon, and the blue circles show the position of the trees.

Now, I seek your help for calculating the number of trees that can be planted on my island.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer N ($3 \leq N \leq 10000$) denoting the number of vertices of the polygon. Each of the next N lines contains two integers x_i y_i ($-10^6 \leq x_i, y_i \leq 10^6$) denoting the co-ordinate of a vertex. The vertices will be given in clockwise or anti-clockwise order. And they will form a simple polygon.

Output

For each case, print the case number and the total number of trees that can be planted inside the polygon.

Sample Input	Output for Sample Input
1 9 1 2 2 1 4 1 4 3 6 2 6 4 4 5 1 5 2 3	Case 1: 8

Note

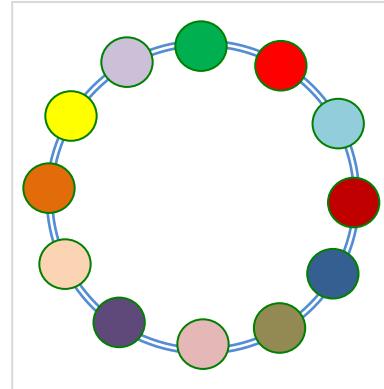
Dataset is huge, use faster I/O methods.

1419 – Necklace

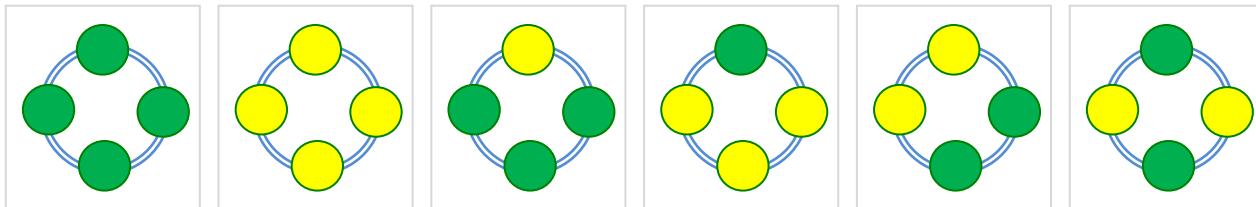
You are a necklace maker. You like this task because it's challenging, fascinating and of course makes a lot of money. Now, you want to make a necklace consisting of n beads. The beads are connected in the following fashion:

- 1) Bead i ($1 < i < n$) is connected with bead $i - 1$ and bead $i + 1$.
- 2) The first bead is connected with the second bead and the n^{th} bead.
- 3) The n^{th} bead is connected with the first bead and the $(n-1)^{\text{th}}$ bead.

Now you have K colors and each bead can be colored using one of the K colors. You have to find the number of possible necklaces you can make using these colors. Two necklaces will be considered same if one can be rotated to another.



For example, say there are 4 beads in the necklace and you have two colors yellow and green, then there are 6 possible necklaces. They are:



Input

Input starts with an integer T (≤ 300), denoting the number of test cases.

Each case starts with a line containing two integers: n and K ($1 \leq n \leq 1000, 1 \leq K \leq 10^9$).

Output

For each case, print the case number and the total number of possible necklaces modulo 1000000007 .

Sample Input	Output for Sample Input
7	Case 1: 6
4 2	Case 2: 3367
5 7	Case 3: 8
5 2	Case 4: 1044
4 8	Case 5: 24
4 3	Case 6: 51
5 3	Case 7: 629
5 5	

1420 – Subsequences forming Strings

Given three strings **A**, **B** and **C** you have to count the number of ways you can construct **C** by combining two **subsequences** from **A** and **B**.

After deleting 0 or more characters from a string we can get its **subsequence**. For example "a", "b", "c", "ab", "ac", "bc", "abc", "" (empty string) are the subsequences of "abc".

Now, suppose there are two subsequences "abc" and "de". By combining them you can get the following strings "abcde", "abdce", "abdec", "adbce", "adbec", "adebc", "dabce", "dabec", "daebc" and "deabc".

Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing three strings **A**, **B** and **C**. The strings will contains only lowercase letters and the lengths of the strings are between **1** and **100** (inclusive).

Output

For each case, print the case number and the number of ways you can construct **C** from the first two strings: **A** and **B** by the above way. The result can be large, so print the result modulo **1000000007**.

Sample Input	Output for Sample Input
2 abc abc abc abbcd bccde abcde	Case 1: 8 Case 2: 18

1421 - Wavio Sequence

Wavio is a sequence of integers. It has some interesting properties:

1. Wavio is of odd length i.e. $L = 2*n + 1$.
2. The first $(n+1)$ integers of Wavio sequence make a strictly increasing sequence.
3. The last $(n+1)$ integers of Wavio sequence make a strictly decreasing sequence.
4. No two adjacent integers are same in a Wavio sequence.

For example **1, 2, 3, 4, 5, 4, 3, 2, 1** is an Wavio sequence of length **9**. But **1, 2, 3, 4, 5, 4, 3, 2, 2** is not a valid wavio sequence. In this problem, you will be given a sequence of integers. You have to find the length of the longest Wavio sequence which is a **subsequence** of the given sequence. Consider the given sequence as:

1 2 3 2 1 2 3 4 3 2 1 5 4 1 2 3 2 2 1

Here the longest Wavio sequence is: **1 2 3 4 5 4 3 2 1**. So, the output will be **9**.

Input

Input starts with an integer **T (≤ 12)**, denoting the number of test cases.

Each case starts with a line containing an integer **N ($1 \leq N \leq 10^5$)** denoting the number of elements in the sequence. The next line contains **N** space separated integers between **-10⁸** to **10⁸**, that form the sequence.

Output

For each case, print the case number and the length of the maximum possible Wavio sequence.

Sample Input	Output for Sample Input
3 10 1 2 3 4 5 4 3 2 1 10 14 1 2 3 2 1 2 3 4 3 2 1 5 4 1 5 1 2 3 4 5	Case 1: 9 Case 2: 7 Case 3: 1

Note

Dataset is huge, use faster I/O methods.

1422 - Halloween Costumes

Gappu has a very busy weekend ahead of him. Because, next weekend is Halloween, and he is planning to attend as many parties as he can. Since it's Halloween, these parties are all costume parties, Gappu always selects his costumes in such a way that it blends with his friends, that is, when he is attending the party, arranged by his comic-book-fan friends, he will go with the costume of Superman, but when the party is arranged contest-buddies, he would go with the costume of 'Chinese Postman'.



Since he is going to attend a number of parties on the Halloween night, and wear costumes accordingly, he will be changing his costumes a number of times. So, to make things a little easier, he may put on costumes one over another (that is he may wear the uniform for the postman, over the superman costume). Before each party he can take off some of the costumes, or wear a new one. That is, if he is wearing the Postman uniform over the Superman costume, and wants to go to a party in Superman costume, he can take off the Postman uniform, or he can wear a new Superman uniform. But, keep in mind that, Gappu doesn't like to wear dresses without cleaning them first, so, after taking off the Postman uniform, he cannot use that again in the Halloween night, if he needs the Postman costume again, he will have to use a new one. He can take off any number of costumes, and if he takes off **k** of the costumes, that will be the last **k** ones (e.g. if he wears costume **A** before costume **B**, to take off **A**, first he has to remove **B**).

Given the parties and the costumes, find the minimum number of costumes Gappu will need in the Halloween night.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with a line containing an integer **N** ($1 \leq N \leq 100$) denoting the number of parties. Next line contains **N** integers, where the **ith** integer **c_i** ($1 \leq c_i \leq 100$) denotes the costume he will be wearing in party **i**. He will attend party 1 first, then party 2, and so on.

Output

For each case, print the case number and the minimum number of required costumes.

Sample Input	Output for Sample Input
2 4 1 2 1 2 7 1 2 1 1 3 2 1	Case 1: 3 Case 2: 4

1423 - Olympic Swimming

'Hurdles' is a common event in athletics where the runners race in running tracks containing hurdles (obstacles). For the race some hurdles are placed evenly spaced along a straight running course. They are positioned so that they will fall over if bumped into by the runner.

The organizers of 20012 Olympic Games are planning to introduce a hurdles event in swimming. For this purpose, some hurdles will be placed in the swimming pool used in the competition. Let's describe the construction of a swimming pool at first. A pool is L meters long and is divided into K equal sized lanes across the width. Each participant swims along his own lane. As events of various lengths take place in the same pool, measurement of length is very critical. So, there is a mark after every meter along the length of the pool starting from 0 in one end. There are some hurdles placed in the pool. A hurdle is placed between two adjacent marks in a lane. Now, the hurdles are not uniformly distributed. For example, in a 5 meter long pool with 2 lanes, the first lane may have 2 hurdles, one between 1m and 2m while another between 3m and 4m. The second lane may have 1 hurdle, between 0m and 1m. For ensuring a fair race, the course must be defined in a way that all the participants have to face same number of hurdles. In other words, you are to select two length marks from the pool so that there is exactly same number of hurdles placed in every lane between these two length marks. You can safely assume that, a race will always start or end in a length mark.

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case starts with a line containing two integers L and K ($1 \leq L \leq 50000, 1 \leq K \leq 30$). Each of the next K lines describes a lane. Each of these lines starts with an integer n ($1 \leq n \leq 5000$) denoting the number of hurdles in the lane, followed by n distinct integers. The i^{th} integer p_i ($0 \leq p_i < L$) denotes that there is a hurdle between length mark p_i and p_i+1 .

Output

For each case, print the case number and the length of the longest race that can take place in the pool.

Sample Input	Output for Sample Input
2 5 2 2 1 3 2 0 2 5 3 3 0 3 4 2 0 3 2 3 4	Case 1: 5 meter(s) Case 2: 3 meter(s)

Note

Dataset is huge, use faster I/O methods.

1424 - New Land

Once upon a time there lived an old wise king. He had only one son. The prince had the idea that he (the prince) was about to be the future king; that's why he became too lazy day by day. This made the wise king a bit worried because his lazy son couldn't be the correct man for the throne. He couldn't sleep; his kingdom was in need of a perfect king, not the lazy king.

One day, while doing his regular works, he found an excellent idea. Next day he called his son. He said that he wanted the prince to have his own kingdom. The prince became very excited. The king continued that the prince can have a land from the king's kingdom, but he should start walking after sunrise and cover a rectangular area before sunset. Then the land would be given to the prince. The lazy prince thought, "It would be an easy task!" That's why he wanted to find the maximum rectangular area in king's land. But there were some rocks in the kingdom, and the prince didn't want any rocks in his new land. He would rather take nothing, but no rocks.

The kingdom can be thought as an **m x n** grid, where **m** is the number of rows and **n** is the number of columns. Each cell in the grid is a small rectangular land whose area is one. If a land contains rock it will be denoted by a **1**, otherwise it will be denoted by a **0**. The prince can walk in the sides of the cells and he can either take a cell (land) or ignore it, but he can't take a part of a cell. Now your task is to find the maximum rectangular area the prince could cover.

Input

Input starts with an integer **T (≤ 4)**, denoting the number of test cases.

Each case starts with a line containing two integers: **m** and **n ($1 \leq m, n \leq 2000$)**. Each of the next **m** lines contains **n** characters (either **0** or **1**) denoting the kingdom.

Output

For each case, print the case number and the maximum rectangular area the prince could take.

Sample Input	Output for Sample Input
2 5 7 0110110 0000010 1000001 0100001 1100010 3 3 001 100 101	Case 1: 12 Case 2: 3

Note

Dataset is huge, use faster I/O methods.

1425 – The Monkey and the Oiled Bamboo

It's time to remember the disastrous moment of the old school math. Yes, the little math problem with the monkey climbing on an oiled bamboo. It goes like:

"A monkey is trying to reach the top of an oiled bamboo. When he climbs up 3 feet, he slips down 2 feet. Climbing up 3 feet takes 3 seconds. Slipping down 2 feet takes 1 second. If the pole is 12 feet tall, how much time does the monkey need to reach the top?"

When I was given the problem, I took it seriously. But after a while I was thinking of killing the monkey instead of doing the horrible math! I had rather different plans (!) for the man who oiled the bamboo.

Now we, the problem-setters, got a similar oiled bamboo. So, we thought we could do better than the traditional monkey. So, I tried first. I jumped and climbed up 3.5 feet (better than the monkey! Huh!) But in the very next second I just slipped and fell off to the ground. I couldn't remember anything after that, when I woke up, I found myself in a bed and the anxious faces of the problem setters around me. So, like old school times, the monkey won with the oiled bamboo.



So, I made another plan (somehow I want to beat the monkey), I took a ladder instead of the bamboo. Initially I am on the ground. In each jump I can jump from the current rung (or the ground) to the next rung only (can't skip rungs). Initially I set my strength factor **k**. The meaning of **k** is, in any jump I can't jump more than **k** feet. And if I jump exactly **k** feet in a jump, **k** is decremented by 1. But if I jump less than **k** feet, **k** remains same.

For example, let the height of the rungs from the ground be 1, 6, 7, 11, 13 respectively and **k** be 5. Now the steps are:

1. Jumped 1 foot from the ground to the 1st rung (ground to 1). Since I jumped less than **k** feet, **k** remains 5.
2. Jumped 5 feet for the next rung (1 to 6). So, **k** becomes 4.
3. Jumped 1 foot for the 3rd rung (6 to 7). So, **k** remains 4.
4. Jumped 4 feet for the 4th rung (7 to 11). Thus **k** becomes 3.
5. Jumped 2 feet for the 5th rung (11 to 13). And so, **k** remains 3.

Now you are given the heights of the rungs of the ladder from the ground, you have to find the minimum strength factor **k**, such that I can reach the top rung.

Input

Input starts with an integer **T** (≤ 12), denoting the number of test cases.

Each case starts with a line containing an integer **n** ($1 \leq n \leq 10^5$) denoting the number of rungs in the ladder. The next line contains **n** space separated integers, $r_1, r_2 \dots, r_n$ ($1 \leq r_1 < r_2 < \dots < r_n \leq 10^9$) denoting the heights of the rungs from the ground.

Output

For each case, print the case number and the minimum possible value of **k** as described above.

Sample Input	Output for Sample Input
2 5 1 6 7 11 13 4 3 9 10 14	Case 1: 5 Case 2: 6

Note

Dataset is huge, use faster I/O methods.

1426 - Blind Escape

Illidan Stormrage, the blind Warcraft hero is thrown to a maze for punishment. Being a blind hero, he cannot see anything but he can sense which direction is north. He memorized the map of the maze when he was a child. But he doesn't know his exact location now.



Assume that the maze is laid out on a grid, and each grid location is either blocked or free. He has four possible moves: **North**, **South**, **East** and **West**. He wants to find the shortest sequence of moves that will guarantee his escape (his initial location can be any free cell in the grid). He is said to be escaped if he is outside the maze and once he is out of the maze, further moves are irrelevant. And of course if he tries to walk into a wall, he will simply stay in the same spot.

As he cannot see anything, for any move, he will only stop if he bumps into a wall or he is out of the maze. Now your task is to help him by finding the shortest sequence of moves.

Input

Input starts with an integer **T** (≤ 200), denoting the number of test cases.

Each case starts with a line containing two integers **M** and **N** ($1 \leq M, N \leq 12$) denoting the number of rows and columns of the grid respectively. Each of the next **M** lines contains **N** characters (either '.' or '#') denoting the maze. '.' means free location and '#' means a wall. Assume that there is at least one free location in the grid.

Output

For each case, print the case number first. If it's impossible to do so, print "**Impossible**". Otherwise, print the shortest possible sequence of moves that guarantees his escape. Show the sequence by the first letters of the moves. As there can be many solutions, print the one that comes lexicographically earliest. See the samples for details.

Sample Input	Output for Sample Input
2 4 8 ##### #...#. .# ##.... # ##.#### 3 4 ### #. .# ###	Case 1: EWSWS Case 2: Impossible

1427 - Substring Frequency (II)

A string is a finite sequence of symbols that are chosen from an alphabet. In this problem you are given a string T and n queries each with a string P_i , where the strings contain lower case English alphabets only. You have to find the number of times P_i occurs as a substring of T .

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case starts with a line containing an integer n ($1 \leq n \leq 500$). The next line contains the string T ($1 \leq |T| \leq 10^6$). Each of the next n lines contains a string P_i ($1 \leq |P_i| \leq 500$).

Output

For each case, print the case number in a single line first. Then for each string P_i , report the number of times it occurs as a substring of T in a single line.

Sample Input	Output for Sample Input
2 5 ababacbacb aba ba ac a abc 3 lightoj oj light lit	Case 1: 2 3 1 4 1 Case 2: 1 1 0

Notes

1. Dataset is huge, use faster I/O methods.
2. If S is a string then $|S|$ denotes the length of S .

1428 – Melody Comparison

Bono, a very famous singer, had always wanted to have a machine to turn the melodies into strings. Last month he earned two million dollars and bought one of those amazing machines, suddenly he began to play some solos to test his new toy. Surprisingly this machine was able to take a piece of song and output a string of lowercase characters from 'a' to 'z'. After playing his two best solos he took the resulting strings **A** and **B** and began to count substrings and repetitions. He is a great singer but is not good in programming at all, that's why he needs you to help him to find how many distinct substrings of **A** that don't contain **B** as a substring.

Input

Input starts with an integer **T** (≤ 25), denoting the number of test cases.

Each case starts with a line containing string **A**. The next line contains string **B**. You can safely assume that the strings are non-empty and none of their length is more than **50000** and they will contain lowercase English alphabets only.

Output

For each case, print the case number and the number of substrings that Bono needs to know.

Sample Input	Output for Sample Input
4 ababa ba ababa a abbabaaaaba aa ababababa bab	Case 1: 3 Case 2: 1 Case 3: 15 Case 4: 5

1429 – Assassin`s Creed (II)

Ezio needs to kill **N** targets located in **N** different cities. The cities are connected by some one way roads. As time is short, Ezio can send a message along with the map to the assassin's bureau to send some assassins who will start visiting cities and killing the targets. An assassin can start from any city, he may visit any city multiple times even the cities that are already visited by other assassins. Now Ezio wants to find the minimum number of assassins needed to kill all the targets.



Input

Input starts with an integer **T** (≤ 70), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **N** ($1 \leq N \leq 1000$) and **M** ($0 \leq M \leq 10000$), where **N** denotes the number of cities and **M** denotes the number of one way roads. Each of the next **M** lines contains two integers **u v** ($1 \leq u, v \leq N, u \neq v$) meaning that there is a road from **u** to **v**. Assume that there can be at most one road from a city **u** to **v**.

Output

For each case, print the case number and the minimum number of assassins needed to kill all targets.

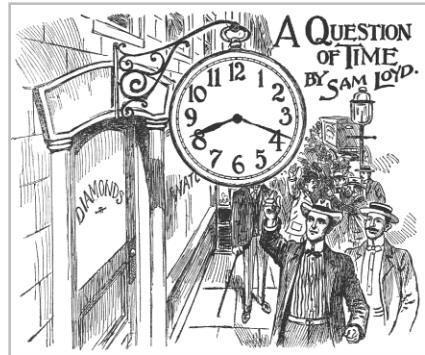
Sample Input	Output for Sample Input
3 5 4 1 2 1 3 4 1 5 1 7 0 8 8 1 2 2 3 3 4 4 1 1 6 6 7 7 8 8 6	Case 1: 2 Case 2: 7 Case 3: 2

Note

Dataset is huge, use faster I/O methods.

1430 – A Question of Time

For this problem, we'll consider a classic puzzle made popular by Sam Loyd, the famous American puzzle author, and recreational mathematician from early 20th century. Mr. Loyd observed a curious thing regarding the signs commonly used in front of jewelry stores in those days, which resembled big watches. The peculiar fact was that these signs were usually created to represent a certain specific time of the day, some minutes past eight. This seemed to occur invariably, in places all over the world, and during centuries since the creation of mechanical clocks.



If one were asked for an explanation for this phenomenon, it's likely that the idea of symmetry would come up, and the fact that the time represented by these signs place the hour hand and the minute hand at the same distance from the 6. Regardless of the psychological reasons behind it, this provides for a nice mathematical challenge.

The original puzzle made by Sam Loyd asked for the exact time represented by these watches, which had to be some time between 8 o'clock and 8:30. However, with the aid of computational resources, it should not be much harder to solve this in general for all hours around the clock, and using any arbitrary hour for the symmetry line.

Write a program that helps you with this extended puzzle. You are presented with a hour hand showing a certain time of the day, which determines the symmetry line, and a range of time (a starting and ending time). Your task is to print all occurrences within that range of time (inclusive) when the hour hand and the minute hand are on opposite sides of the symmetry line, and at the same distance from it.

Input

Input starts with an integer **T** (≤ 20000), denoting the number of test cases.

Each test case starts with three time representations in the form **HH:MM:SS**. The first of these is the hour hand time showing the symmetry line, the second of them is the starting time, and the last one is the ending time, and it is guaranteed that the starting time doesn't come chronologically after the ending time. The times are separated by single spaces. You may assume that **0 ≤ HH < 12, 0 ≤ MM, SS < 60**.

As you may have noticed, for simplicity, the hour 12 will be replaced by hour zero.

Output

For each case, print the case number and the number of answers for the case in a single line. Then print all answers in chronological order, each one in its own line.

Always use 2 digits to represent the amount of hours, minutes, and the integer part of the seconds. To avoid accuracy problems, if the number of seconds is not an exact integer, then it has to be printed as a mixed number with its fractional part in the form p/q , where p and q are coprimes. See the samples below for more details.

As in the input, in the output you should not use hour 12, but hour zero instead.

Sample Input	Output for Sample Input
5	Case 1: 1
06:00:00 08:00:00 09:00:00	08:18:27 9/13
06:00:00 00:00:00 00:45:15	Case 2: 1
09:33:41 02:18:27 03:20:13	00:00:00
01:00:00 01:18:27 02:00:00	Case 3: 2
09:30:00 03:44:17 05:00:57	02:23:38 8/13 03:19:01 9/13 Case 4: 1 02:00:00 Case 5: 1 04:13:50 10/13

Notes

- 1) Be careful about the 4th test case, for the resulting time - 02:00:00, there is no trailing space.
- 2) For the 5th case, the red hour hand showing in the picture determines the symmetry line.



1431 – The Party for the Rich

A party has been arranged for n richest people of city Richtown. They have set some strange rules such that only rich people can join the party. The rule is that, when a person enters the party, he is given two cards - **entrance** and **exit**. They are represented as (x, y) , where x is the integer written on the entrance card and y is the integer written on the exit card. No two persons can have entrance (or exit) cards with same integers written on them. When a person exits the party, he has to pay the dollar equivalent to the absolute difference of the integers in his entrance and exit cards.

Though they were rich, they were very clever. They planned that they will exchange their cards such a way that the total money paid by them is as low as possible. Any people can exchange cards multiple times and with multiple persons. But the entrance cards are distinguishable from the exit cards, so entrance cards are exchangeable with entrance cards only and same rule suffices for exit cards. For example, suppose there are three people having cards with $(1, 5), (7, 3), (8, 10)$, then if they don't exchange cards, they have to pay $|1-5| + |7 - 3| + |8 - 10| = 10\$$. But if they exchange them to $(1, 3), (7, 5), (8, 10)$ then they need to pay $|1-3| + |7-5| + |8-10| = 6\$$.

But there is one problem, each person must pay at least **K\$**, otherwise the organizers will suspect that they have been cheating. So, your job is to help them find the solution where they have to pay as less as possible without creating any suspicion.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n ($1 \leq n \leq 10000$)** and **K ($0 \leq K \leq 2$)**. Each of the next **n** lines contains two integers (between **1** and **20000**) denoting the integers written on the entrance and exit cards respectively for **ith** person.

Output

For each case, print the case number and the minimum amount of money they need to pay. If its impossible to do so, print "**impossible**".

Sample Input	Output for Sample Input
2 3 1 1 1 7 3 8 10 1 2 10 9	Case 1: 10 Case 2: impossible

Note

Dataset is huge, use faster I/O methods.

1432 – Overlapping Sticks

N sticks are randomly dropped on the 2D cartesian plane. You have to find the number of stick pairs that are in overlapping position. Two sticks are said to be in overlapping position if they have infinitely many intersection points. A stick is defined by two distinct points (x_1, y_1) and (x_2, y_2) which denote the end points of the stick.

Input

Input starts with an integer T (≤ 5), denoting the number of test cases.

Each case starts with a line containing an integer N ($1 \leq N \leq 10^5$). Each of the next N lines contains four integers x_1, y_1, x_2, y_2 ($-10^6 \leq x_1, y_1, x_2, y_2 \leq 10^6$) forming a segment.

Output

For each case, print the case number and the number of stick pairs in overlapping position.

Sample Input	Output for Sample Input
2 4 1 0 5 0 5 0 8 0 4 0 6 0 0 0 10 0 2 1 1 5 5 5 1 1 5	Case 1: 5 Case 2: 0

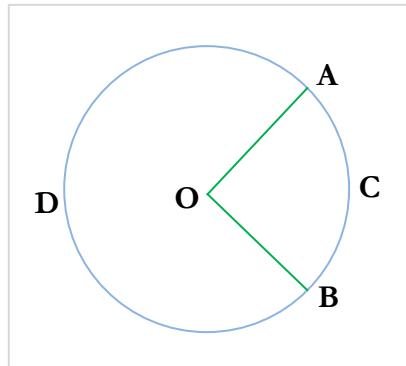
Note

Dataset is huge, use faster I/O methods.

1433 - Minimum Arc Distance

You all probably know how to calculate the distance between two points in two dimensional cartesian plane. But in this problem you have to find the minimum arc distance between two points and they are on a circle centered at another point.

You will be given the co-ordinates of the points **A** and **B** and co-ordinate of the center **O**. You just have to calculate the minimum arc distance between **A** and **B**. In the picture, you have to calculate the length of arc **ACB**. You can assume that **A** and **B** will always be on the circle centered at **O**.



Input

Input starts with an integer **T** (≤ 100), denoting the number of test cases.

Each case starts with a line containing six integers $O_x, O_y, A_x, A_y, B_x, B_y$ where (O_x, O_y) indicates the co-ordinate of **O**, (A_x, A_y) denote the co-ordinate of **A** and (B_x, B_y) denote the co-ordinate of **B**. All the integers will lie in the range [1, 10000].

Output

For each case, print the case number and the minimum arc distance. Errors less than 10^{-3} will be ignored.

Sample Input	Output for Sample Input
5 5711 3044 477 2186 3257 7746 3233 31 3336 1489 1775 134 453 4480 1137 6678 2395 5716 8757 2995 4807 8660 2294 5429 4439 4272 1366 8741 6820 9145	Case 1: 6641.81699183 Case 2: 2295.92880 Case 3: 1616.690325 Case 4: 4155.64159340 Case 5: 5732.01250253

1434 - Patch Quilt

We will now try another problem inspired by one of Sam Loyd's puzzles.

A group of children made a lovely patch quilt that they gave as a gift to their teacher. Knowing their teacher is a puzzle aficionado, the kids created the quilt with a very particular feature: the names of everyone who contributed to the gift are hidden in the patchwork.

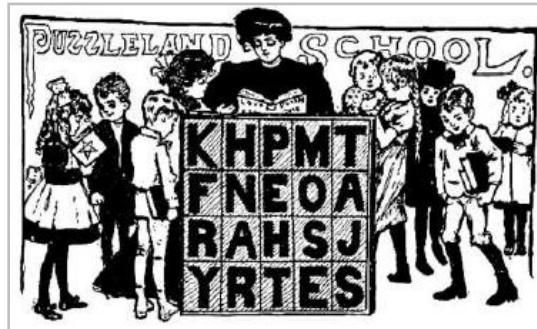


Figure 1: A present from her students

The teacher was happily surprised by the thoughtful gift and kept it in the classroom, where it is displayed and continues to pose a challenge to anyone who gazes upon it. The rules of the puzzle are simple:

- 1) To find any name, you can start at any position, and at each step you may move in any direction, including diagonals.
- 2) Any position in the puzzle can be used more than once. You can even choose not to move at some steps, to use the same letter in succession.
- 3) For every name that is hidden in the puzzle, there is only one valid way to find it (there are no multiple solutions for any of the names).

You know the list of names of all the students. Your task is to identify their locations inside the quilt.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a line containing two integers **R** and **C ($1 \leq R, C \leq 30$)** denoting the number of rows and columns of the quilt respectively. The following **R** lines contain **C** letters each, representing the contents of the puzzle. All letters are uppercase.

The next line contains an integer **N ($1 \leq N \leq 20$)**, denoting the number of students. Each of the next **N** lines contains a name, containing uppercase letters. The length of each name will be at least **2**, and at most **15**. You can safely assume that if a certain name is found in the puzzle, there will only be one possible way to discover it.

Output

For each case, print the case number in a single line. Then, for each name given in the input, print if it is found in the puzzle or not.

If a name is found, print the message "**NAME found:**" followed by a description of its location, given in the following way:

- 1) First print the coordinates of the first letter in the form **(r,c)** where $1 \leq r \leq R$ and $1 \leq c \leq C$.
- 2) Then, for each additional letter, print its location relative to the previous letter. Use the letters **U, D, L, R** for up, down, left and right respectively. For diagonals, use the following combinations: **UL** for up-left, **UR** for up-right, **DL** for down-left, and **DR** for down-right. To indicate no movement at all, use an asterisk **(*)**.
- 3) Make sure to use a comma and a single space between elements in the list. Notice, however, that there must be no spaces after the last move.

If a name is not found, simply print "**NAME not found**". For more details on the format of the output, please refer to the sample below.

Sample Input	Output for Sample Input
1 4 5 KHPMT FNEOA RAHSJ YRTES 4 JAMES JOHN HANNAH MARIE	Case 1: JAMES found: (3,5), U, UL, DL, DR JOHN found: (3,5), UL, DL, UL HANNAH found: (3,3), L, U, *, D, R MARIE not found