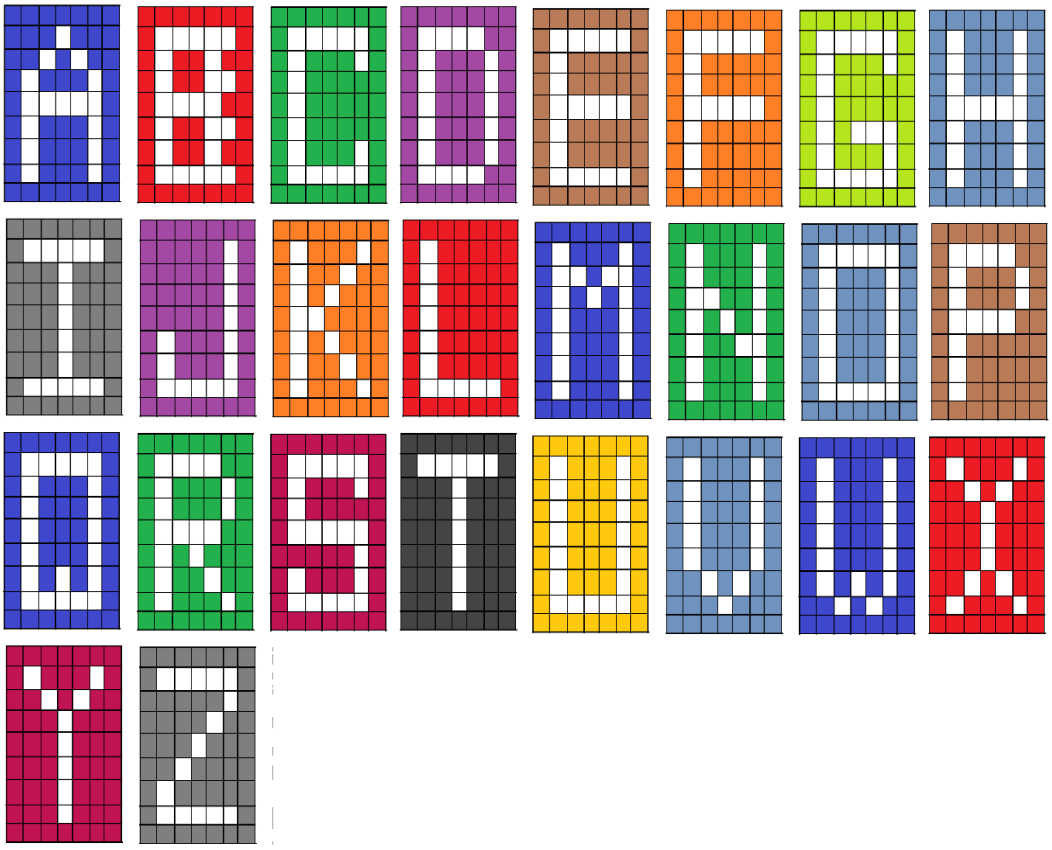
Alphabet

Every letter in alphabet or character can be represented by **2D** grids of **7X5** dimension. Computer graphics uses this technique to save and print characters for long time. We can represent each English letter (capital **A-Z**) like the image below.



Human minds are very much pattern sensitive that they can easily recognize different characters from these grids even though the grid can be full of garbage.

```
A.R..GGGGG
.R.RC..G..
R...R..G..
RRRRR..G..
R...R..G..
R...R..G..
R.X.RGGGGG
```

We can easily detect that there are an **A** (composed of character **R**) and an **I** (composed of character **G**) pattern in the above grid. But can you write a computer program to detect them?

Human can also detect pattern with some noise like

```
..R.R          .....
```

```
.R.R.          .....
R...R          ..R..
RRRRR          .R.R.
R...R          RRRRR
R...R          R...R
R...R          R...R
```

But you should not detect them as A. You should be deterministic. Also size of each letter should be same as given in the grids above.

Given an **NXM** grid of printable characters (without whitespace and newline), find how many English letter (capital **A - Z**) patterns are there. A pattern will be recognized as a letter if and only if all of it's characters are same and placement of these characters are same compared to each other as given in the reference grids. As the grid maintains **8-adjacency** [each **cell(i, j)** has **8** adjacent cells as **cell(i-1,j-1), cell(i-1,j), cell(i-1,j+1), cell(i,j-1), cell(i,j+1), cell(i+1,j-1), cell(i+1,j)** and **cell(i+1,j+1)**], if the pattern composing character is **X** then there should not be any **X** character adjacent to these pattern composing characters, otherwise the pattern will not be a valid one (see the example above and the last sample input for more clarity).

Input

First line of input will be a positive integer **T (T ≤ 50),** the number of test cases, followed by **T** test cases. First line of each test case contains two space separated integers **N** and **M (7 ≤ N ≤ 100, 5 ≤ M ≤ 100)**, number of row and number of column of the given grid respectively. Each of the next **N** lines will contain **M** printable characters without whitespace and newline (ASCII **33** to **126**).

Output

For each test case, print the test case number followed by a new line. Then print either **"No Alphabet Found!"**, without the quotes, if there is no letter pattern found in the grid or print the characters in increasing order from **A** to **Z** and the number of times it was found in the grid in separate lines. If the count of a character pattern is **0**, then ignore the character to print. See the sample output for more clarity.

Sample Input	Sample Output
3 7 10 ..R..GGGGG .R.R...G.. R...R..G.. RRRRR..G.. R...R..G.. R...R..G.. R...RGGGGG 12 14 ZZRZZGGGGGZZZZ ZRZRZZZGZZZZZZ	Case 1: A 1 I 1 Case 2: I 1 T 1 Case 3: No Alphabet Found!

```
RZZZRZZGZZZZZZ
RRRRRZZGZZZZZZ
RKKKRKKGKaaaaa
RKKKRKKGKKKaKK
RKKKRGGGGGKaKK
KKKKKRKKKKKaKK
KaabaaKKKKKaKK
KKKbKKKKKKKaKK
KKKbKKKKKKKaKK
KKKbKKKKKKKKK
7 10
..R..RRRRR
.R.R...R..
R...R..R..
RRRRR..R..
R...R..R..
R...R..R..
R...RRRRRR
```

Browser History

Tiger X has developed a new web browser. His web browser has the following property:

- Browser stores a browsed URL, if it is not available in the cache. So it will have every visited URL stored into its cache.
- When someone writes a string **S** in the web browser, it collects all the cached URL's that has prefix **S**. Matched URL's are displayed in **lexicographic** order as suggestion.

You are given his browsing histories in the following format:

S D

S – is the string he wrote in the browser.

D – is the **1-based** index of the URL, which he chose to visit from the browsers suggested URL list. If the value of **D** is **0**, that means he visited the URL **S** directly.

Now he wants you to write a program that will tell the **minimum possible** number of URL **N** in browser cache, after browsing all the URL's.

Input

First line of input is the number of test case **T (T ≤ 20)**.

First line of each test case is a number **Q (Q ≤ 5 * 10^4)** denoting the number of browsing done by tiger X. Each of the next **Q** line contains string **S (1 ≤ length(S) ≤ 5 * 10^4)** and **D (0 ≤ D ≤ 10^5)** separated by a single space. The characters in string **S** will be from lowercase English alphabet (**a-z**).

Note: *For each test case,* $X \leq 5 * 10^5$, $X = \sum_{i=1}^{Q} length(S_i)$. *Input file size is at most* ***3MB****.*

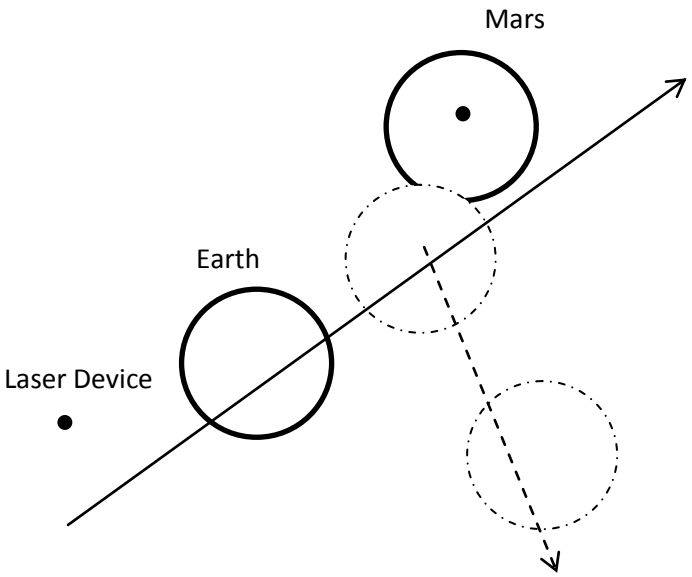
Output

For each test case print the test case number followed by the number **N**. See sample output for output format.

Sample Input	Sample Output
2 1 a 0 3 a 3 ab 1 abc 2	Case 1: 1 Case 2: 3

Crazy Minion

After disappearing the Moon, Gru targeted something bigger. He removed all planets from the solar system except Earth and Mars. Then he made two devices. First one is stopping device and second one is laser device. The stopping device can freeze any planet if he knows the exact position of center of that planet. After being frozen by the stopping device no planet can move. A frozen planet starts moving again only when the laser device is used on that planet. Otherwise it can't be displaced by any other forces. Laser device is too small; you can consider it as a point. It is controlled remotely. If anybody pulls the trigger of the laser device after selecting a planet it emits a laser ray form the position of the device to the direction of the center of that planate. After that the planet starts traveling **V** units/second in a straight line according to the direction of laser. The laser device only affects the selected planet. It does not affect other planet, even if it is in the path of the laser. He froze Earth and Mars by stopping device. The point (x_D,y_D,z_D) containing laser device is outside of all planets. Consider Earth as solid sphere having R_E radius and center (x_E,y_E,z_E). Consider Mars as also a solid sphere having R_M radius and center (x_M,y_M,z_M).



One day a crazy minion was playing with that device and pulled the trigger after selecting the Earth. After that Earth starts traveling V units/second to the direction of laser. Gru wants to stop the moving Earth. Now Gru needs to know the exact position of the Earth. Help Gru to find the current position of center of the earth. Meanwhile Earth travels T seconds. Unfortunately Earth can collide with Mars on the way. If so, only the direction of Earth will be changed but velocity will remain constant. Though position of Mars will remain unchanged whether collision happens or not. It is an elastic collision. There is no loss of energy and no extra forces. No damage will happen. You need to help Gru to find the current co-ordinate of the center of Earth. Any pair of planets will never overlap.

Input

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

Each case contains four lines. First line of the input contains four integers R_E, x_E, y_E, z_E. Second line contains four integers R_M, x_M, y_M, z_M. Third line contains three integers x_D, y_D, z_D. Fourth line contains two integers **V** and **T**. Absolute value of all co-ordinates are not more than 10^4. Distance between the both centers will be more than R_M+R_E. **(0 < V, T, R_M, R_E ≤ 1000)**.

Output

For each case, print the case number and the center of the co-ordinate of Earth after traveling **T** seconds. There should be at least 3 decimal points. Absolute or relative error below 10^{-3} will be ignored. Follow the sample output format. There is no such test case for which any of the output co-ordinates could be `-0.000`.

Sample Input	Sample Output
2 1 0 0 0 1 5 7 0 -2 -2 0 1 15 1 0 0 0 1 5 7 0 0 0 -11 1 1000	Case 1: 10.607 -0.607 0.000 Case 2: 0.000 0.000 1000.000

Dare to Review

Mr. Tom just appeared at an examination. The examination is based on MCQ (Multiple Choice Questions). One has to fill up the correct choice in an OMR sheet. There were total **30** questions in the exam. All questions carry equal weight. For each correct answer one gets **4** points while for each incorrect answer **1** point is deducted (or **-1** point is added). Questions those are not answered have no effect on the score. For example if anyone answers 10 questions and 8 of them are correct then his score will be **(8 * 4 – 2 * 1) = 30**. As Mr. Tom does not want to reduce his score by answering incorrectly, he only answered those he is almost sure. Thus he answered **18** questions and expected a good score. But when the result is published, he found that he only got **59**. He was disappointed and began to think about it. Soon he realizes that the score **59** is an impossible score for **18** questions answered. So he appealed for a review. And yes, there was some malfunction in the OMR reader, so the authority later fixed his score.

Now you are to do the same task but in more general manner. Given an examination description, number of questions answered and published score. You need to determine, whether the score is possible or not.

Input

The first line of input will contain the number of test cases **T** (**T ≤ 2500**). Each of the next following lines will have 5 integers **Q, P, D, A** and **S** (**0 ≤ A ≤ Q, 0 < Q ≤ 10^6, 0 ≤ P, D ≤ 20, -10^8 ≤ S ≤ 10^8**). Here **Q** is the number of total questions, **P** is the score for each correct answer, **D** is the score to be deducted for each incorrect answer, **A** is the number of question answered and **S** is the published score.

Output

For each input print the case number and a string containing either "**It may happen**" if the score is possible for the mentioned situation or "**Review needed**" (quotes are for clarity) if the score is impossible under the conditions. See sample input output for exact format.

Sample Input	Sample Output
5 30 4 1 18 59 30 4 1 21 59 100 5 2 75 0 100 5 2 75 -150 100 10 5 100 1000	Case 1: Review needed Case 2: It may happen Case 3: Review needed Case 4: It may happen Case 5: It may happen

Even vs Odd Divisor

Every number has some divisors that perfectly divide the number. For example, the divisors of **6** are **1, 2, 3** and **6**. Some of the divisors are even and some are odd. Mr. Mario likes the even divisors. So he prefers the numbers for which the number of even divisors is greater than the number of odd divisors. He defined a term even beautiful number with respect to **n (n > 2)**, if the number of even divisors is greater than or equal to **n** times the number of odd divisor. For example, **8** is an even beautiful number with respect to **3** because it has **3** even divisors against **1** odd divisor while **6** is not as it has **2** even and **2** odd divisors. You are given two numbers **A** and **B** and another number **n (A ≤ B)**. You need to find the number of even beautiful numbers with respect to **n** between **A** and **B** (inclusive).

Input

The first line of input will denote the number of test cases **T (T < 400000)**. Each of the following **T** lines will contain two non negative integers **A, B** and another number **n (0 ≤ A ≤ B ≤ 10^9, 2 ≤ n ≤ 100)**.

Output

For each input, first print the case number followed by one line containing the number of even beautiful numbers within **A** and **B**, inclusive.

Sample Input	Sample Output
2 3 10 3 17 23 3	Case 1: 1 Case 2: 0

Fallen Rook

On an **N X N** chessboard where **(1, 1)** is the leftmost top and **(N, N)** is the rightmost bottom, Mr. Fallen wants to place **N** rooks such that no two rooks attack each another. Two rooks attack each other if they are on the same row or the same column. After placing all the **N** rooks on the board, Mr. Fallen looks for interesting cells and marks them with **"?"**. A cell **(R, C)** is considered **interesting** if there is a rook on cell **(R_1, C)** where **R_1 > R** and there is a rook on cell **(R, C_1)** where **C_1 > C**.

Mr. Fallen, as always, has some extra conditions to be happy. He wants to see exactly **K "?"**s on the board.

Given **N** and **K**, please tell Mr. Fallen how many arrangements of **N** non attacking rooks are possible such that there will be exactly **K** interesting cells (marked with **"?"**)on an **N X N** chessboard.

Input

The first line of the input contains an integer **T (T ≤ 100,000**) denoting the number of test cases. Each of the following **T** lines has two space separated integers **N** and **K (1 ≤ N ≤ 5000, 0 ≤ K ≤ 5000)**

Output

For each input, print the output in the format, **"Case C: X"** (quote for clarity), where **C** is the case number and **X** is the number of different arrangements modulo **1,00,00,00,007**.

For exact output format please check sample input/output section.

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

Game of MJ 2

M and J are playing a game. M gives J an integer ***V*** (a large one) and asks J to find the sum of digits of ***V***. To M's surprise J gives the correct answer every time and J gives the answer really fast. After a few rounds of the game, M asks J, "How can you solve it so fast?" J replies with a smile, "I have written a program, which solves it for me."

Hearing this, M comes up with a tougher game for J to solve. Let's see how J can solve this, M thinks.

M now gives an integer ***V*** with length ***N***. Then M changes ***V*** by the following command: ***"i j d"***. This means "Change the digits in ***V*** from position ***i*** to ***j*** to ***d***", where ***0*** is the position of the most significant digit of ***V*** and ***N-1*** is the position of the least significant digit. After each of these commands, J needs to calculate the sum of digits for ***U***, where ***U = 2V***.

For example, let ***V*** = 1234. If M's command is "0 2 2", ***V*** becomes 2224. ***U*** = 2***V*** = 4448. So sum of digits of ***U*** = 4 + 4 + 4 + 8 = 20, which is the answer J should provide.

M is successful. J cannot solve this game even with the help of a computer. Now J's reputation is at stake and you have to save J. Please help J.

Input

First line of input contains one integer ***T (T<21)***, the number of test cases. Next there will be T cases. Each case begins with the value of ***N (N<100001)*** and ***Q (Q<50001)***, where ***Q*** is the number of commands M will give. ***Q*** lines will follow. Each line will contain three integer, ***i, j (0<= i, j < N)*** and ***d (0<= d <=9)***. You may assume that, initially ***V*** is 0 (zero).

Output

For each case, print one line "Case C" in one line, where ***C*** is the case number. In subsequent lines, write the sum of digits of ***U*** after M gives each of the commands.

Sample Input	Sample Output
1 4 3 0 2 2 0 1 3 1 3 4	Case 1 12 16 30

How Many Relation Set

In this problem, you will be given an undirected graph **G=(V,E)** where **V** is the set of vertices and **E** is the set of edges. You have to count number of distinct relation set.

Let **R** be the relation set. Below is the definition of **R**:

1. Let, **R** = **{S,C}** where **S** and **C** are two sets of vertices and there is no vertex in common between **S** and **C** (**S** and **C** are mutually exclusive and **S** ⊆ **V and C** ⊆ **V**).
2. There will be no edge between any two vertices of **S**.
3. Every vertex in **S** must have an edge with every vertex in **C** and every vertex in **S** must have edges only with vertices in **C**.
4. **S** and **C** each must contain at least one element.

Input

First line of the input contains T, denoting there will be T test cases (T <= 5) cases. The first line of each test case contains two integers **N** and **M** (**1 ≤ N ≤ 1000,1 ≤ M ≤ 500000**) **,** where **N** is the number of vertices in the graph and **M** is the number of edges respectively. Then follows **M** lines where each line contains two integers, **u** and **v** (1 ≤ u ≤ N, 1 ≤ v ≤ N, u≠ v) separated by spaces, denoting that there is an edge between u and v. Edge will not be repeated.

Output

For each test case, first print the case number, followed by the number of distinct relation sets. As the output can be large, so print output modulo 1,000,000,007.

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

Imperative Strings

You are given a string ***s*** (set of lowercase letters) with some rules and find that whether the string is derived using the given rules. The rules are produced using capital and small letters. The rules are only in two formats as below:

$$A \rightarrow BC \text{ or } A \rightarrow b$$

If the right hand side of the rule contains two letters then the letters must be capital letter and if the right hand side of the rule contains one letter then the letter must be small letter. And Left hand side is always capital letter. One letter may occur in more than one rule. For example:

A->BC
B->CA
C->a
A->b
B->c

Using therse rules, you can derive ***aabaa*** from letter *A* using following steps:

$A \rightarrow \boldsymbol{BC} \rightarrow \boldsymbol{CA}C \rightarrow \boldsymbol{a}AC \rightarrow a\boldsymbol{BC}C \rightarrow a\boldsymbol{CA}CC \rightarrow a\boldsymbol{a}ACC \rightarrow aa\boldsymbol{b}CC \rightarrow aab\boldsymbol{a}C \rightarrow aaba\boldsymbol{a}$

Your task is to find whether the given string is derived from the given starting symbol using the given rules.

Input

First line of the input set contains an integer **T (T<10)** denoting the number of test cases. Each test case starts with an integer **n (1 ≤ n ≤ 50)** denoting the number of rules. Next **n** lines contain the rules. Next line contains a string ***s*** $(\mathbf{1} \leq |\boldsymbol{s}| \leq \mathbf{50})$**.** s is consisted of lowercase English characters only. Next line contains a capital letter denoting the starting symbol.

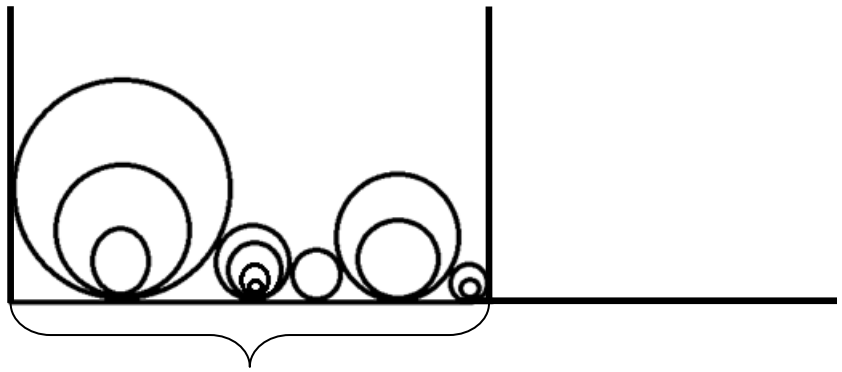
Output

For every test case, print one line “Case x: y” where x denotes the case number starting from 1 and y denotes “Yes” if the string is derived from the given starting symbol otherwise “No” (without the quotes).

Sample Input	Sample Output
1 5 A->BC B->CA C->a A->b B->c Aabaa A	Case 1: Yes

Jumbled Pipes

Have you ever visited town hall market in Mohammadpur? I guess you have or will soon, because you are at present sitting at a location very near to it. A certain businessman, named Tiny, works at a pipe shop at town hall. His shop is very small and he has only one table to display his pipes. The pipe provider has arrived to provide pipes for his shop. They will give **n** pipes to Tiny's shop. As Tiny has only one small table to put all the pipes he has, he must be clever enough to optimally use the table for displaying pipes (Otherwise he doesn't get enough space to place his 6 inch display smart phone!). Tiny has requested for n different pipes of various radii in a specific order (he is very careful about the order of the pipes). The pipe provider also gives the pipes in the specific order that Tiny requested.



Your job is to put the pipes in the table one after one. So simple heh? But no! Tiny has allowed you to put a pipe inside another pipe provided that the outer pipe's radius is strictly greater than the inner pipe. Moreover, you cannot put a small pipe inside a large pipe, if the large pipe already contains any smaller pipes that means, you cannot put a pipe of radius **4** inside a pipe of radius **6** that already contains another pipe of radius **3**. As Tiny's shop is tiny, you do not have any extra table to store the pipes while they come. That means, when a new pipe comes, you can either put it within a previously kept pipe or you can put it beside the previous pipes on the table (not above any other pipe!). When putting it beside the previous pipes, you must ensure that its touching point with the table is placed to the right of any previous pipes' touching point with the table. The first pipe should touch the leftmost wall with what the table is attached. Your goal is to minimize the width of the table used to contain all the pipes according to the constraints. Width used is defined by the distance between the leftmost wall of which the table is attached and the rightmost vertical line that touches the last pipe.

Note that, when a pipe is placed inside another pipe, the inner pipe must touch the touching point where the outer pipe touches the table/its enclosing pipe. It should be according to the gravity! The width of pipe walls is negligible.

Input

The first line of input contains **T (T ≤ 200)**, the number of test cases. The T cases follow. Each test contains an integer **N (1 ≤ N ≤ 18)**, denoting the number of pipes in order. Followed by N positive integers. Each integer is **≤ 1000**.

Output

For each case, first print the test case number, followed by the minimum width of the table used to contain all the pipes as described above, up to 6 decimal places. Your output should have a **relative error ≤ 10^{-4}**

Sample Input	Sample Output
3 5 5 3 4 2 1 3 5 3 3 5 5 4 3 2 1	Case 1: 15.745967 Case 2: 15.745967 Case 3: 10.000000

Kid's Sum

Given two integers **A** and **B** your job is to find the quantity $f(A, B)$**,** where

$$f(A, B) = \sum_{i=A}^{B} \left\lfloor \frac{i}{SOD(i)} \right\rfloor$$

Where $\lfloor x \rfloor$ denotes the largest integer less than or equal to **x**. And $SOD(y)$ denotes the sum of digits of the number y, so SOD(35) = 3+5 = 8, SOD (100) = 1+0+0 = 1.

Input

The first line of contains an integer **T** (**≤ 10000**) which denotes the number of test cases. Each of the following **T** lines contain two space separated integers **A and B** (**1 ≤ A ≤ B <10^9**).

Output

For each test case, print the case number followed by the required quantity.

Sample Input	Sample Output
3 8 13 80 120 10000 800000000	Case 1: 24 Case 2: 698 Case 3: 8163185581355389

LaMurgi

There is a country called Strangeland that contains **N** cities. The Government collects tax from companies of these cities while it donates some help to companies of some other cities in order to develop them. In other word, each company in a particular city has some fixed negative (paying tax), positive (receive donation) or zero financial help from the Government. Interestingly, this financial help follows an interesting rule. There are some directed relationship between some pairs of cities and if there is a relation from city **a** to city **b**, then each companies in city **b** will have 1 unit more financial help from the Government, i.e. if companies of city **a** receives help of **t** units then companies of city **b** will receive help of (**t + 1**) units. The relationships will be such that there are no direct or indirect multiple relationships between a pair of cities. Government also has a helping factor **s** for each city that get multiplied to the financial help of that city. So if the helping factor of a city is s and it gets t units of help from government, the total help value of this citiy would be s*t. This factor is fixed for each city. Now the Government is preparing the budget and going to announce the financial help for companies in the cities maintaining the constraints mentioned above. The Government has already decided the possible minimum and maximum financial help t_{min} and t_{max} and the helping factor **s** for companies from each of the cities, but the exact figure is yet to be announced.

LaMurgi is a popular fast food company in Strangeland. It has shops in each of the cities. The owner LaMurgi is a dishonest and powerful person. He has good relationship with some policy maker and he is trying to influence them to announce the financial help such a way that maximizes the total help for LaMurgi. But he cannot change the multiplicative factor. The policymakers gave him the decided minimum and maximum help for each of the cities and asked him to submit a plan. Now he hired you to calculate the maximum help LaMurgi can get.

Input

The first line of the input file will denote the number of test cases **T (T ≤ 30)**. Then there will be **T** cases. Each case will start with two integers **N (1 ≤ N ≤ 100,000)** and **M (0<=M<N)** denoting the number of cities and number of relations respectively. Each of the following **M** lines will contain two integers $\mathbf{a_i}$ and $\mathbf{b_i}$ (**0 ≤ a_i,b_i ≤ N-1**) denoting a relation from $\mathbf{a_i}$ to $\mathbf{b_i}$. The next line will contain **N** space separated integers $\mathbf{S_i}$ **(0 ≤ S_i ≤ 1000000**) denoting the multiplicative factor for city **i**. The next line will contain **N** space separated integers $\mathbf{min_i}$ **(-1000000 ≤ min_i ≤ 1000000**) denoting the minimum limit on financial help for city **i**. The next line will contain the **N** space separated integers $\mathbf{max_i}$ **(min_i ≤ max_i ≤ 1000000**) denoting the maximum limit on financial help for companies in city **i**.

Output

For each of the test cases print the case number and either the text “Impossible” if it is not possible to allocate help to cities according to the constraints or the maximum possible help LaMurgi can get in a single line. See sample input output section for exact format of output.

Sample Input	Sample Output

2 3 2 0 1 1 2 2 2 2 2 2 2 4 4 4 3 2 0 1 1 2 2 2 2 2 2 2 3 3 3	Case 1: 18 Case 2: Impossible