Problem A. Game of Peace

Program:	`numbers.(cpp|java)`
Input:	`numbers.in`
Balloon Color:	`Pink`

Bob has learned a new magic trick that needs a very special preparation. Once he masters the trick he will be able to bring peace to the world, but if he fails, the world will be destroyed.

The preparation is performed as follows: There are two containers, initially one is empty and the other one has $\boldsymbol{X}$ marbles. Bob has a Marble Cloning Machine, it clones the marbles in the container with the larger number of marbles, then pours the new clones into the other container (e.g. if the two containers have 7 and 4 marbles, after the cloning step they will have 7 and 11 marbles). The machine does this cloning operation exactly $\boldsymbol{M}$ times. However, there is a bug in the machine, after it performs $\boldsymbol{N}$ cloning operations ($\boldsymbol{N} \leq \boldsymbol{M}$), it will add $\boldsymbol{Y}$ extra marbles to the container with the larger number of marbles. Then the machine will continue normally with the cloning operation exactly $\boldsymbol{M} - \boldsymbol{N}$ times.

During the cloning operations, if both containers have the same number of marbles, any of them can be considered the one with the larger number of marbles.

Now, the bug in Bob's machine is threatening to destroy the world. But his nerdy friend Alice told him that she knows how to fix it. All he has to do is to calculate the greatest common divisor of the sizes of the two containers after the cloning machine is done. Can you help Bob save the world?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ ($1 \leq \boldsymbol{T} \leq 1{,}000$) representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case will consist of a single line, containing 4 integers separated by a single space $\boldsymbol{X}$, $\boldsymbol{N}$, $\boldsymbol{Y}$ and $\boldsymbol{M}$ ($1 \leq \boldsymbol{X}$, $\boldsymbol{Y} \leq 1{,}000$) ($0 \leq \boldsymbol{N} \leq 70$) ($\boldsymbol{N} \leq \boldsymbol{M} \leq 100{,}000$) which are the numbers as described above.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a space then the greatest common divisor of the sizes of the two containers after the machine is done.

Examples

numbers.in	stdout

2
4 3 6 5
5 1 15 2
	Case 1: 2
	Case 2: 5

Note

In the first sample test case, the number of marbles in each container will be the following after each step: (4, 0), (4, 4), (4, 8), (12, 8), (18, 8), (18, 26), (44, 26). The greatest common divisor of 44 and 26 is 2.

Problem B. Let's Play Tawla

Program:	`tawla.(cpp|java)`
Input:	`tawla.in`
Balloon Color:	`Blue`

Tawla is the Arabic name of the game Backgammon. In Tawla, 2 players alternate playing two 6-face dice. Each die (singular of dice) face represents a number from 1 to 6 through black dots carved on that face.

Tawla professionals give the numbers 1 to 6 special naming, believed to be adapted from another language. This special naming is as follows:

- 1: "Yakk"
- 2: "Doh"
- 3: "Seh"
- 4: "Ghar"
- 5: "Bang"
- 6: "Sheesh"

Tawla professionals have this habit of saying the dice number after they throw the dice, in order to have some sort of game commentary. The higher number is said first.

Some examples:

- A dice throw of 1 and 2 is: "Doh Yakk"
- A dice throw of 3 and 5 is: "Bang Seh"
- A dice throw of 6 and 4 is: "Sheesh Ghar"

If you know more about Tawla, you would know that a double (2 dice producing the same number) does not follow this rule. For some unknown reason, it doesn't rhyme to say: "Yakk Yakk" or "Doh Doh". The following are their special names:

- A 1-1 dice pair is said: "Habb Yakk"
- A 2-2 dice pair is said: "Dobara"
- A 3-3 dice pair is said: "Dousa"
- A 4-4 dice pair is said: "Dorgy"
- A 5-5 dice pair is said: "Dabash"
- A 6-6 dice pair is said: "Dosh"

One exception to all the above rules is the pair: 5-6 (or 6-5), this one is called "Sheesh Beesh" and not "Sheesh Bang"! As you may have expected, this is for some unknown reason too.

Write a program that translates dice numbers to Tawla words.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ $(1 \leq \boldsymbol{T} \leq 100)$ representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case will consist of a single line, containing 2 integers separated by a single space $\boldsymbol{a}$ and $\boldsymbol{b}$ $(1 \leq \boldsymbol{a}, \boldsymbol{b} \leq 6)$ representing the number of black dots carved on the top face of each die.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a space then the Tawla words describing the given dice numbers.

Examples

tawla.in	stdout

3
1 2
2 3
3 4
	Case 1: Doh Yakk
	Case 2: Seh Doh
	Case 3: Ghar Seh

Problem C. Feast Coins

Program:	`coins.(cpp|java)`
Input:	`coins.in`
Balloon Color:	`Red`

Last feast the young princess received way too many coins. Since she is very young, she doesn't know the value of each coin, if you give her a coin with the value 5 or a coin with the value 1, she will consider them both as just 1 coin, regardless of the value.

However, she can notice that the coin with value 5 doesn't look the same as the coin with value 1, and she will be happy if she has the same number of coins of each value. Otherwise, she will not be happy.

She has a lot of coins of different values, and she needs some subset of these coins such that the sum of their values should be exactly $\boldsymbol{S}$, and the number of coins of each value in this subset should be the same. Can you help her to count the number of different ways to do this?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ ($1 \leq \boldsymbol{T} \leq 100$) representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case starts with a line containing two integers separated by a single space $\boldsymbol{S}$ and $\boldsymbol{N}$ ($1 \leq \boldsymbol{S} \leq 5{,}000$) ($1 \leq \boldsymbol{N} \leq 50$) representing the total required sum and the number of different values of coins, respectively. Followed by $\boldsymbol{N}$ lines, each one will contain two integers separated by a single space $\mathbf{V}_i$ and $\mathbf{C}_i$ ($1 \leq \mathbf{V}_i, \mathbf{C}_i \leq 5{,}000$) representing the value of a coin and the number of coins the princess has with this value, respectively. For each test case, all values of $\mathbf{V}_i$ will be distinct.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a space then the number of different ways to make the total sum $\boldsymbol{S}$ as described above. Two ways are considered different if any coin value does not appear the same number of times in both ways.

You can assume that the result will always fit in a 64-bit signed integer.

Examples

coins.in	stdout

2
10 2
2 2
6 1
10 4
1 10
2 10
3 10
4 10
	Case 1: 0
	Case 2: 5

Note

In the first test case, the only way to make the sum 10, is to use the following subset of coins (2, 2, 6), but this isn't valid because there are 2 coins with value 2 and 1 coin with value 6.

In the second test case, the following are the 5 different ways: (1, 1, 1, 1, 1, 1, 1, 1, 1, 1), (2, 2, 2, 2, 2), (2, 2, 3, 3), (1, 1, 4, 4), (1, 2, 3, 4).

Problem D. Wedding Selfie

Program:	`balance.(cpp|java)`
Input:	`balance.in`
Balloon Color:	`Green`

At his wedding, Daryl took a beautiful selfie with his bride. In fact, the selfie was such a unique photograph that Daryl and his wife decided to hang it on their new home's wall. The selfie was shaped as a convex polygon with $\boldsymbol{N}$ vertices. Merle, Daryl's lousy brother, offered to hang the selfie for them. Merle used $\boldsymbol{N}$ nails, one for each vertex, to hang the selfie on the wall. However, only the first nail he used was tightened properly. The other nails were pretty loose and did not stay fixed for long. So, after some time Daryl found the beautiful selfie dangling awkwardly on his wall, hanging only from the tightened nail. Your job is to find out the new position of the polygon's vertices after the selfie finally settles.

You will be given the original positions of the vertices of the polygon, in counter clockwise order, and starting with the one Merle tightened properly. For simplicity, the tightened vertex will always be at $(0, 0)$. When the polygon starts dangling, it will be able to rotate freely about this vertex. The polygon will settle only when the area of the polygon to the right of the Y-axis is equal to the area to the left of the Y-axis. If there are multiple such positions, the image will settle in the position that maximizes the polygon area below the X-axis. Can you help Daryl to figure out the final position of the vertices of the picture after it settles?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ $(1 \leq \boldsymbol{T} \leq 100)$ representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case starts with a line containing an integer $\boldsymbol{N}$ $(3 \leq \boldsymbol{N} \leq 100)$ representing the number of vertices. Followed by $\boldsymbol{N}$ lines, each line will contain 2 integers separated by a single space, $\mathbf{X}_i$ and $\mathbf{Y}_i$ (-1,000 $\leq \mathbf{X}_i$, $\mathbf{Y}_i \leq$ 1,000) representing the coordinates of the i-th vertex. You can assume that no 3 consecutive vertices will lie on the same line, and the first vertex (which has the tightened nail) of each test case will be at (0, 0).

In a convex polygon, all interior angles are less than 180 degrees. It's guaranteed that all given polygons will be convex.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by $\boldsymbol{N}$ lines, each line should contain 2 values separated by a single space, $\mathbf{X}_i$ and $\mathbf{Y}_i$, rounded to exactly 6 decimal places after the decimal point, representing the new coordinates of the i-th vertex after the selfie had settled, in the same order as given in the input.

Make sure not to print -0.000000, instead print 0.000000.

Examples

balance.in	stdout

1
4
0 0
1 0
1 1
0 1
	Case 1:
	0.000000 0.000000
	-0.707107 -0.707107
	0.000000 -1.414214
	0.707107 -0.707107

Problem E. Connect the Cells

Program:	`cells.(cpp|java)`
Input:	`cells.in`
Balloon Color:	`Black`

"Connect the Cells" is a very famous game, you can find a version of the game on any mobile device.

The game is played on a board of N rows with N cells in each row. Each cell is either empty or colored. We represent an empty cell with '0' and colored cell with a digit from '1' to '9'. For each color that exists on the board, there will be exactly 2 cells with that color. Your task is to connect every pair of cells of the same color with each other, without leaving any empty cells.

Two cells are adjacent to each other if they share an edge, vertically or horizontally. Cells colored with the same color in each input grid will never be adjacent to each other.

Here is how you can connect 2 cells of the same color: Let's say you have a pen for each color and you will use it to color the cells, and once this pen touches any cell it will color it. You will start by putting the pen on one of the 2 cells, and keep moving it from its current cell to another adjacent cell, through empty cells, until you finally move it to the second cell with the same color. Once the pen enters the second cell of that color, these 2 cells are considered connected and you should stop using this pen, and start connecting the other colors (if there are still some left). The pen can not leave the board or color the same cell more than once.

The only exception when the pen can be on an already colored cell, when it's on the starting cell or the ending cell of its color, and it can be on any of them only once.

Can you write a program to solve this game?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T ($1 \leq T \leq 100$) representing the number of test cases. Followed by T test cases. Each test case will start with a line containing an integer N ($3 \leq N \leq 8$) representing the size of the board. Followed by N lines, each one consists of N digits. Each digit represents a cell, with '0' meaning it is an empty cell. It's guaranteed that there will be at least one valid solution for each test case, and all input boards will satisfy all the conditions mentioned above.

For each test case, if the board contains X distinct colors, they will be named using the digits from 1 to X, inclusive (there will be at least one color and at most 9 colors in each test case).

Output

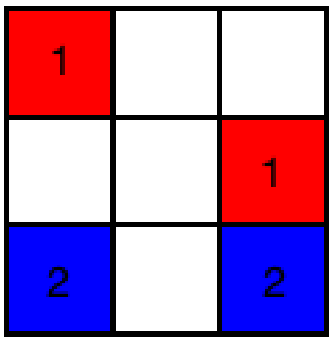
For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by N lines, each line should contain N characters. The j-th character in the i-th line represents the direction you used to exit from the j-th cell in the i-th row in the grid ('U' for up, 'R' for right, 'D' for down, 'L' for left and 'X' if it was the last cell you entered after connecting the cells of its color). If there are multiple solutions, print any of them.

Examples

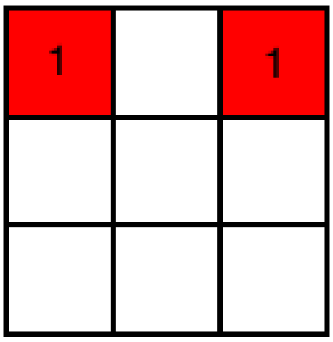
cells.in	stdout
2	Case 1:
3	RRD
100	RDX
001	URX
202	Case 2:
3	DRX
101	DUL
000	RRU
000	

Note

Here are the 2 sample cases:

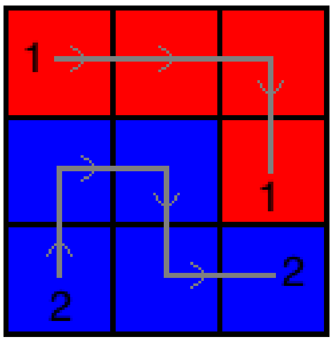


First Case

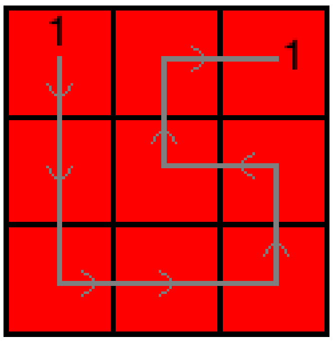


Second Case

And here is a valid solution for each one:



First Case



Second Case

Problem F. Bike Sharing

Program:	`bike.(cpp|java)`
Input:	`bike.in`
Balloon Color:	`Purple`

The city of Sharm El Sheikh is planning to open a bike sharing service in the city. They will build $\boldsymbol{N}$ bike stations at strategic points in the city. Each station will have a capacity $\boldsymbol{C}$, meaning that at most $\boldsymbol{C}$ commuters may take a bike from a station in the morning, and at most $\boldsymbol{C}$ commuters may return a bike to a station in the evening. These numbers are independent (i.e. a station can still accept $\boldsymbol{C}$ bikes in the evening even if no bikes were taken from it in the morning).

You are in the planning department of this service, and you decided to do some market research to make sure the service is profitable. Here is what you found out: There are exactly $\boldsymbol{M}$ different groups of possible commuters that would like to use your service. Each group consists of $\boldsymbol{P}$ individuals, each individual in the same group wants to use the service in the same way: they all want to grab a bike from the a certain start station $\boldsymbol{St}$ in the morning, and return it to a certain (possibly the same) end station $\boldsymbol{En}$ in the evening, and each individual is willing to pay exactly $\boldsymbol{X}$ in return for this service. Since commuters need their bike for the whole day, no bike can be used by multiple commuters.

The number of bikes in a station may not satisfy all commuters. In this case you are free to pick the subset of commuters that you can satisfy that will pay the biggest amount in total (you don't need to pick the whole group, it's okay to pick a subset of the same group).

Now armed with the above information, you have to choose the best possible value for $\boldsymbol{C}$ (possibly zero). Of course having a larger capacity comes with a cost. The cost of having a capacity $\boldsymbol{C}$ is $\boldsymbol{D} \times \boldsymbol{C}$. Can you figure out the maximum profit (total revenue − total cost) this service can generate?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ ($1 \le \boldsymbol{T} \le 50$) representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case starts with a line containing 3 integers separated by a single space $\boldsymbol{N}$, $\boldsymbol{M}$ and $\boldsymbol{D}$ ($1 \le \boldsymbol{N} \le 50$) ($1 \le \boldsymbol{M} \le 250$) ($1 \le \boldsymbol{D} \le 100{,}000$) representing the number of stations, the number of groups of possible commuters, and the cost per extra bike in all stations, respectively. Followed by $\boldsymbol{M}$ lines, each line will contain 4 integers $\boldsymbol{P}$, $\boldsymbol{St}$, $\boldsymbol{En}$ and $\boldsymbol{X}$ ($1 \le \boldsymbol{P} \le 100{,}000$) ($1 \le \boldsymbol{St}, \boldsymbol{En} \le \boldsymbol{N}$) ($1 \le \boldsymbol{X} \le 100{,}000$) representing the number of individuals in the group, the start and end of their commute, and the amount each individual is willing to pay, respectively.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a space then the maximum profit.

Examples

bike.in	stdout

2
2 3 3
10 1 2 2
10 1 1 2
10 2 2 2
2 3 5
10 1 2 10
10 1 1 2
10 2 2 2
	Case 1: 10
	Case 2: 50

Problem G. Special Christmas Tree

Program:	`tree.(cpp|java)`
Input:	`tree.in`
Balloon Color:	`White`

Christmas is coming and everyone is building a Christmas tree, and you are no exception. However you are special and you want to build a special one. You decided to build a binary tree and to hang its root from the ceiling. For this problem a binary tree can be defined as a collection of connected nodes. The topmost node is called the root. Every node in the tree might have 0, 1 or 2 other nodes hanging from it, called children. Nodes with no children are called leaves. And every node has exactly 1 parent, except the root has no parent.

You bought a decoration pack containing some items and you want to use all of them to decorate all the leaves of the tree. You are limited by the height of your room, so the tree can not be longer than it. The height of tree is the number of edges on the path from the root to the farthest leaf.

Note that each leaf must be decorated by exactly 1 item (and each item can decorate exactly 1 leaf), and you must use all items.

Can you find the most special tree? Tree X is more special than tree Y, if X has more nodes than Y.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ ($1 \leq \boldsymbol{T} \leq 10{,}000$) representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case will consist of a single line, containing 2 integers separated by a single space $\boldsymbol{H}$ and $\boldsymbol{L}$ ($0 \leq \boldsymbol{H} \leq 1{,}000{,}000{,}000$) ($1 \leq \boldsymbol{L} \leq 1{,}000{,}000{,}000$) ($1 \leq \boldsymbol{L} \leq 2^{\boldsymbol{H}}$) representing the maximum possible height and the number of leaves of the tree, respectively.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a single space then the number of nodes in the most special Christmas tree that has exactly $\boldsymbol{L}$ leaves and height at most $\boldsymbol{H}$.

Examples

tree.in	stdout

2
3 2
3 3
	Case 1: 7
	Case 2: 9

Problem H. Messed up Pictures

Program:	`picture.(cpp|java)`
Input:	`picture.in`
Balloon Color:	`Yellow`

Your friend had a lot of awesome pictures on her computer, until someone messed them up. She found out that most of her image files were either cropped and/or rotated by 90 degrees, or not changed at all. Luckily, she still remembers the original combined set of aspect ratios of all her images.

An aspect ratio for an image of dimensions (A, B) is (A / G, B / G) where G is the greatest common divisor (GCD) of A and B.

Since we don't care about the image contents, a rotation operation is swapping the width and the height. And a crop operation is selecting a smaller sub-rectangle (with integer dimensions) inside the original image and its sides should be parallel to the sides of the original image.

Your job is to help her to identify a possible original dimensions of each image. Given the dimensions of the messed up images, along with the possible aspect ratios, output a possible original dimensions of each image as well as how many operations were done to mess it up.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ ($1 \le \boldsymbol{T} \le 100$) representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case starts with a line containing an integer $\boldsymbol{M}$ ($1 \le \boldsymbol{M} \le 100$) representing the number of possible aspect ratios followed by $\boldsymbol{M}$ lines, each line will contain 2 integers $\boldsymbol{Rx}$ and $\boldsymbol{Ry}$ ($1 \le \boldsymbol{Rx} \le \boldsymbol{Ry} \le 100$) representing the i-th aspect ratio (for any given aspect ratio, the GCD of $\boldsymbol{Rx}$ and $\boldsymbol{Ry}$ will always be 1), all aspect ratios in the same test case will be distinct. The next line will contain an integer $\boldsymbol{N}$ ($1 \le \boldsymbol{N} \le 100$) representing the number of messed up images, followed by $\boldsymbol{N}$ lines, each line will contain 2 integers $\boldsymbol{X}$ and $\boldsymbol{Y}$ ($1 \le \boldsymbol{X}, \boldsymbol{Y} \le 100$) representing the dimensions of the i-th messed up image.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by $\boldsymbol{N}$ lines, each line should contain 3 integers separated by a single space $\boldsymbol{W}$, $\boldsymbol{H}$ and $\boldsymbol{K}$ representing a possible original dimensions for the i-th messed up image (the ratio of $\boldsymbol{W}$ and $\boldsymbol{H}$ should be one of the given ratios) and the number of operations used to mess it up ($\boldsymbol{K}$ should be 0 if no changes were made, 1 if it was cropped or rotated, 2 if it was cropped and rotated).

For each image, if there are multiple solutions, print the one that minimizes $\boldsymbol{W} \times \boldsymbol{H}$. If there are still multiple solutions, print the one that minimizes $\boldsymbol{W}$. If there are still multiple solutions, print the one that minimizes $\boldsymbol{K}$.

Examples

picture.in	stdout

1
1
3 4
5
3 4
4 3
2 4
4 2
5 6
	Case 1:
	3 4 0
	3 4 1
	3 4 1
	3 4 2
	6 8 1

Problem I. Equivalent Passwords

Program:	`password.(cpp|java)`
Input:	`password.in`
Balloon Color:	`Gold`

Yesterday you arrived at the hotel, and you kept all your valuable stuff in your room's safe. Unfortunately, you forgot the password. But you have a very long list of passwords (each password is at most 5 digits), and you are sure that your password is one of them.

The safe will consider some passwords equivalent. Two passwords A and B are considered equivalent, if they are of the same length, and |A[i] - B[i]| is the same for all possible values of i, where X[i] is the i-th digit of X and |Y| is the absolute value of Y.

You will go through the list of passwords in the given order. For each password, you will do the following:

1. If the same password or any of its equivalent passwords were typed before, skip this password.
2. Otherwise, type this password into the safe.
3. If it's the correct password (or any of its equivalent passwords), the safe will open and you will stop any further processing.

Now given the list of all passwords, you would like to know, in the worst case scenario, what is the maximum number of passwords you will have to type?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ ($1 \leq \boldsymbol{T} \leq 50$) representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case starts with a line will containing an integer $\boldsymbol{N}$ ($1 \leq \boldsymbol{N} \leq 100{,}000$) representing the number of passwords, followed by $\boldsymbol{N}$ lines, each one will contain a non-empty string of at most 5 digits (from '0' to '9'), representing a password (might contain leading zeros).

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a space then the maximum number of passwords you will have to type.

Examples

password.in	stdout
3 3 000 111 222 4 1111 123 214 2222 3 43434 54545 45454	Case 1: 1 Case 2: 2 Case 3: 2

Note

In the first test case: all passwords are equivalent to each other. This means that the first password will open the safe for sure.

In the second test case:

- If the first password is the correct one, you will type 1 password.
- If the second password is the correct one, you will type 2 passwords.
- If the third password is the correct one, you will type 2 passwords (because the second password is equivalent to the third one).
- If the fourth password is the correct one, you will type 1 password (because the first password is equivalent to the fourth one).

In the third test case:

- If the first password is the correct one, you will type 1 password.
- If the second password is the correct one, you will type 1 password (because the first password is equivalent to the second one).
- If the third password is the correct one, you will type 2 passwords. Even though the third password is equivalent to the second password, the second password was skipped, and therefore you should type the third password.

Problem J. Mario and Evil Toad

Program:	`mario.(cpp|java)`
Input:	`mario.in`
Balloon Color:	`Cyan`

Mario is trying to save Princess Peach from the evil Bowser. But at the end of every level, a small mushroom guy called Toad tells him that she is at a different castle. Mario suspects that Toad is actually evil, and keeps sending Mario to the furthest castle possible just for fun.

Mario's world can be modeled as a tree with $\boldsymbol{N}$ nodes (a tree of $\boldsymbol{N}$ nodes is a connected graph with $\boldsymbol{N}$-1 edges, nodes are numbered from 1 to $\boldsymbol{N}$), where each node represents a castle, and each edge represents a level. Mario starts his journey at the root of the tree, and travels around the tree (possibly passing the same nodes and edges several times) looking for the Princess in castles, according to the directions of Toad. It takes Mario a certain amount of time to cross a level (and it is the same in both direction).

Toad picks $\boldsymbol{K}$ distinct castles (the root isn't one of them), and Mario must visit them in the exact given order (Mario must start and end his tour at the root). Mario always takes the shortest path when going from one node to another. What is the maximum possible duration of Mario's journey if Toad had picked the worst possible $\boldsymbol{K}$ castles?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$ ($1 \leq \boldsymbol{T} \leq 100$) representing the number of test cases. Followed by $\boldsymbol{T}$ test cases. Each test case starts with a line containing 2 integers $\boldsymbol{N}$ and $\boldsymbol{K}$ ($2 \leq \boldsymbol{N} \leq 1{,}000$) ($1 \leq \boldsymbol{K} \leq 100$) ($1 \leq \boldsymbol{K} \leq \boldsymbol{N}$ - 1) representing the total number of castles in the tree and the number of castles Mario has to visit. Followed by $\boldsymbol{N}$-1 lines, each line describes the nodes 2 to $\boldsymbol{N}$. Each line will contain 2 integers $\mathbf{P}_i$ and $\mathbf{C}_i$ ($1 \leq \mathbf{P}_i \leq \boldsymbol{N}$) ($0 \leq \mathbf{C}_i \leq 100{,}000$) representing the parent of the i-th node (i starts from 2 here) and the time it takes to pass the edge connecting the current node to its parent. Node 1 is the root of the tree.

Output

For each test case print a single line containing "Case n:" (without quotes) where n is the test case number (starting from 1) followed by a space, followed by the maximum duration for Mario's journey.

Examples

mario.in	stdout

3
3 1
1 5
1 3
4 2
1 5
1 3
3 2
5 3
1 5
1 3
3 2
3 10
	Case 1: 10
	Case 2: 20
	Case 3: 46

Note

For the third test case, one of the possible lists that Toad could have given Mario is following: 5, 2, 4.