2020 Florida High School Programming Series
End of Year Contest

May 2, 2020

Problem/Filename	Problem Name
toiletpaper	Toilet Paper Reselling
usernames	Generating Usernames
assignment	Group Assignment
boardgame	Board Game
grading	Class Grades
raffle	School Raffle
xray	Alternate X-Ray
jump	Jump Conveyor
roads	Roads? Where We Are Going We Don't Need Roads.
ornaments	Christmas Ornaments
scavenger	Scavenger Hunt
essential	Essential Road Work

Toilet Paper Reselling
Filename: *toiletpaper*
Time Limit: *3 seconds*

People resell items all the time. During last Christmas season, there was a video on YouTube about a guy who drove around to all the Walmarts in three states buying Millennial Monopoly and selling each copy for double of Walmart's selling price. Since there was a shortage, his copies of the game sold and he made a killing.

However, in a time of emergency you cannot raise prices of essential items. It is illegal and you can be fined for price gouging. But that hasn't stopped people from doing it. Third party selling websites like Amazon and eBay have started banning sellers from offering hiked up prices on items such as toilet paper.

So Sleazy McGee decided to visit every Walmart and Target in Florida and bought all the toilet paper he could find. His average price per roll was $1. His plan is to sell them online for $5. Unknownst to him, if he sells 100 rolls, his account will go into review and he won't be able to sell any more rolls past the first one hundred.

The Problem
Given how many toilet paper rolls Sleazy McGee purchases, calculate the amount of profit he will earn.

The Input
The first line of input will contain a single positive integer, n ($n \leq 1{,}000$), representing the number of input cases to process. The input cases follow, one per line. Each input case will have a single positive integer, t ($t \leq 1{,}000{,}0000$), representing the number of rolls purchased.

The Output
Output the amount of profit from buying this much toilet paper as an integer. A loss is shown as a negative profit.

Sample Input	**Sample Output**
5	20
5	80
20	400
100	390
110	-73
573	

Generating Usernames
Filename: *usernames*
Time Limit: *3 seconds*

This is the first individual contest for the Florida High School Programming Contest. Normally, usernames are generated based on the school name and sent to the team sponsor. However this system won't work for an individual contest.

In an attempt to use participant's names to generate usernames without identifying participants, usernames will be created in the following manner: the full first name of the participant will be followed by the first two letters of the participant's last name, followed by the underscore character ('_'), followed by the school's initials.

For example, Kyle Dencker (problem author) went to Boynton Beach Community High School. So Kyle's username would be KyleDe_BBCHS.

The Problem
Given a person's first and last names followed by their school, create their username.

The Input
The first line of input will contain a single positive integer, n ($n \leq 100$), representing the number of input usernames to process. The input cases follow, one per line. Each input case starts with an integer, m ($3 \leq m \leq 20$), representing the number of words separated by a space on the line. The first word will be the first name of the contestant, followed by their last name. Both names will have at least 2 characters. The next m-2 words will be the name of the school. All words on the line are proper nouns, so they will have a capital letter starting each one.

The Output
For each input case, on a line by itself, output the generated username.

Sample Input
```
4
7 Kyle Dencker Boynton Beach Community High School
6 Arup Guha Winter Park High School
7 Glenn Martin William R. Boone High School
6 Sharon Barak Timber Creek High School
```

Sample Output
```
KyleDe_BBCHS
ArupGu_WPHS
GlennMa_WRBHS
SharonBa_TCHS
```

Group Assignment
Filename: *assignment*
Time Limit: *3 seconds*

In his AP Computer Science Principles class, Mr. Dencker has his students program in pairs. In the age of digital learning, he decided to keep up the tradition, but you have found out that as long as you or your partner completes the work, you will end up getting a 100% on the assignment.

When Mr. Dencker pulls the grades, he gets a report in a string of 1s and 0s. The '1' represents a student completing the work, and '0' they haven't. He has asked you to develop a program that calculates each group's grade. In addition, as much as you think it is unfair, Mr. Dencker always rounds his grades *down* to the nearest integer.

The Problem
Mr. Dencker has a report of how each group did on this week's assignments. Help him calculate the final grade for each group. The grade will be a simple average of the number of completed assignments by the group, over the total number of assignments.

The Input
The first line will have a single positive integer, g ($1 \leq g \leq 10{,}000$) representing the number of groups needed to grade. Each group will have two lines of equal length. Each line will represent a student's completed work on the given assignments. The length of each line will have a length of 1 to 30 inclusively.

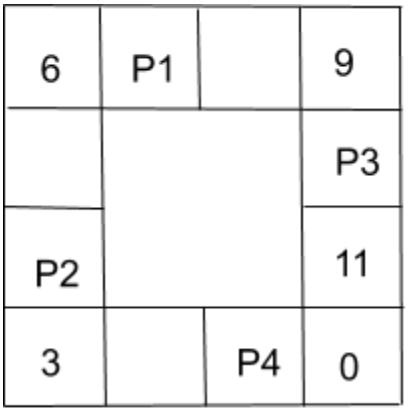
The Output
Output the final grade that the group would get that week as an integer.

Sample Input
```
3
101100101
010011010
00011
01101
101011
101011
```

Sample Output
```
100
80
66
```

Board Game
Filename: *boardgame*
Time Limit: *1 second*

Spend more time at home these days, you've decided to invent a board game, so, of course, you don't get bored!!! Your game has *n* squares arranged in s cyclic order, numbered 0 through *n*-1. Right after a player moves to square *n*-1, when they advance one more square, they go back to square 0. However, you don't particularly like multiple players on the same square, so you've decided that if a player is to advance some number of squares, all squares that are passed with other players don't count. Just like most board games, all players initially start on square 0, and to advance, they roll a pair of dice and advance the number of squares shown as the sum of the dice rolls. But, after each player's initial move, she will never share a square with another piece, including other players who haven't moved yet. Consider the game shown below (in the middle of the game) on a board with $n = 12$:



In this board, Player 1 (P1) is on square 7, Player 2 (P2) is on square 4, Player 3 is on square 10 and Player 4 is on square 1. Suppose that it's Player 3's turn and she rolls a sum of 7 on her two dice. The squares that would count are: 11, 0, 2, 3, 5, 6 and then 8, which is where she would move for the turn. Specifically, she jumped/skipped squares 1, 4 and 7 because these were occupied by Player 4, Player 2 and Player 1, respectively.

Counting manually has been too much of a hassle these days. Write a program to calculate where each player will end up after a series of rolls in a game.

The Problem
Given the number of squares on the game board, the number of players in the game, the number of moves that have been made, as well as the roll for each of those moves, calculate which game square each player is on after the completion of those moves.

The Input

The first line of input will contain a single positive integer, c ($c \leq 20$), representing the number of input cases to process. The input cases follow. The first line of each input case contains three space separated positive integers, n ($6 \leq n \leq 100$), representing the number of squares on the game board, p ($2 \leq p \leq 10$, $p < n$), representing the number of players in the game, and t ($0 \leq t \leq 100$), the number of turns played in the game so far. The second line of each input case contains t space separated integers, representing the sum of the dice of each of the t dice rolls, in the order that they were made. Players move in order, with Player 1 going first, then Player 2, etc. After player p, Player 1 goes again.

The Output

For each case, output p lines, each with a single integer, such that the i^{th} line is the square number (0-based) where the i^{th} player is located on the game board after the t turns.

Sample Input

```
2
12 4 7
2 3 8 10 4 9 7
10 5 2
5 6
```

Sample Output

```
7
4
8
1
5
7
0
0
0
```

Sample Explanation

In the first sample, player 1 first moves to square 2. Then, player 2 moves to square 4 (moving 3 spaces). Then player 3 moves 8 squares to square 10 (skipping squares 2 and 4). Player 4 moves 10 squares, skipping squares 2, 4 and 10, landing on square 1, after completing a lap. Player 1 goes again advancing 4 squares, skipping over square 4, landing on square 7. Player 2 advances 9 squares, skipping over squares 7, 10 and 1, landing again on square 4. On the last move, Player 3 advances 7 squares, as described on the previous page, skipping over squares 1, 4 and 7 to land on square 8.

In the second sample, only 2 turns occur before we print out where each player is. Player 1 advances to square 5, while player 2 advances 6 squares, skipping over square 5, landing on square 7. The other three players are on the starting square, square 0.

Class Grades
Filename: *grading*
Time Limit: *3 seconds*

School has seemed to go digital now. With distance learning, not only are students learning something new, so are their teachers. Teachers are having trouble determining what things to keep the same and what to change in the virtual setting.

However, what hasn't changed is the amount of paperwork that needs to be completed by teachers each day. One day Mr. Dencker's administrator sent an email asking for a list of grades students are earning in his class. She wants in sorted order from lowest grade to highest.

Mr. Dencker started this task but because of the horrible web interface it is taking too much time. The web interface only allows him to manually sort by swapping students that are next to each other in a list. Knowing that his administrator won't look at it very closely, he decides to get creative and put the student with the lowest grade at the beginning of the list, and the student with the highest at the end of the list, knowing that his administrator won't look anywhere in between.

Can you write Mr. Dencker a program that will tell him the minimum number of swaps he will need to make for each class so that the minimum grade appears first in the list and the maximum grade appears last in the list?

The Problem
Given a list of grades, find the minimum number of swaps of consecutive grades necessary to have the lowest grade first, and the highest grade last in the list.

The Input
The first line will have a single positive integer, n ($1 \leq n \leq 1{,}000$) representing the number of classes he needs help with. Information for each class follows. The first of each of these lines will have a line with a single integer, m ($1 \leq m \leq 10{,}000$), which will represent the number of grades to be sorted. On the next line there will be m grades, in their original order, separated by spaces. Each grade will be between 0 and 110, inclusive. Online grading comes with its perks, which include up to 10 extra credit points!!!

The Output
On a line by itself, for each input case, output the minimum number of swaps of consecutive grades needed to make sure that the smallest grade is first and the largest is last.

Sample Input

```
2
4
80 70 90 85
6
90 90 80 80 70 60
```

Sample Output

```
2
8
```

School Raffle
Filename: *raffle*
Time Limit: *1 second*

Your school is raffling off a number of items. Luckily, you know exactly when the opportunity to submit raffle tickets for items closes, so that you can be the last person to submit tickets, knowing exactly how many others have submitted tickets for each item. You are only allowed to submit 1 ticket per item. Naturally, you want to maximize the expected value of the items you win in the raffle. Write a program to calculate this maximum expected value.

The Problem

Given the number of items in the raffle, the number of raffle tickets you are going to submit, as well as both the value (in dollars) and the number of tickets already submitted for each raffle item, calculate the maximum expected value of the items you win in the raffle.

The Input

The first line of input will contain a single positive integer, c ($c \leq 20$), representing the number of input cases to process. The input cases follow. The first line of each input case contains two space separated positive integers, r ($1 \leq r \leq 100$) and s ($1 \leq s \leq r$), representing the number of items in the raffle, and the number of raffle tickets you are submitting, respectively. The following r lines will contain two integers each: v ($1 \leq v \leq 10000$) and t ($0 \leq t \leq 10000$), the value of a raffle item in dollars and the number of raffle tickets already bought for that item, respectively. Each line represents the information for one item in the raffle.

The Output

For each case, output a single line with a floating point number rounded to 2 decimal places representing the maximum expected value of the items you win at the raffle, assuming you buy your raffle tickets for the appropriate items. Note: the input cases will be such that adding or subtracting 10^{-6} from the actual expected value won't change its value rounded to 2 decimal places.

Sample Input

```
2
3 2
10 3
50 27
3 1
4 1
5 0
100 9
22 10
91 8
```

Sample Output

```
4.29
10.11
```

Alternate X-Ray
Filename: *xray*
Time Limit: *3 seconds*

In order to combat dangerous diseases doctors developed a new technology - the X-Ray. An X-Ray is a geometric concept, like a line, or a ray. X-Rays are centered on a point and have four perpendicular rays coming out of it, forming the shape of an X (this is where the name X-Ray comes from). Each ray also forms a 45° angle with the X and Y axes. When someone is hit by the X-Ray, they are instantly cured of any disease.

The Problem

The doctors have n patients loitering in the hospital. They would like to activate the X-Ray once in order to cure as many people as possible. They would like to know the maximum number of people they can cure with one activation, assuming they locate the center of the X-Ray appropriately.

The Input

The first line of input will contain a single positive integer, n ($n \leq 20$), representing the number of input cases to process. The input cases follow. The first line of each input case contains a single positive integer, p ($1 \leq p \leq 1000$) representing the number of patients. Then, there will be p lines, each with two space separated integers, x_i and y_i ($|x_i|, |y_i| \leq 10^6$) representing the position of the i^{th} patient. No two patients occupy the same position.

The Output

For each case, output a single line with a single integer representing the answer to the input case.

Sample Input

```
2
3
1 1
2 2
3 3
6
2 7
4 9
1 8
1 6
5 4
2 8
```

Sample Output

```
3
5
```

Note: It is not necessary for the X-Ray to be centered at a point with integer coordinates!

Jump Conveyor

Filename: *jump*

Time Limit: *5 seconds*

Emma is a creative child who got bored during the quarantine season. She set up n trampolines in a line inside the living room, each aimed in a particular direction. The i^{th} trampoline has value v_i. When Emma jumps to position i, if there is a trampoline, she will jump to position $i+v_i$. Of course, this could result in infinite jumping.

The Problem

Emma's younger brother, Oliver, is very concerned for his sister's safety. He would like to determine how many trampolines are unsafe for Emma to jump on initially. A trampoline is unsafe if it leads to infinite jumping.

The Input

The first line of input will contain a single positive integer, t ($t \leq 20$), representing the number of input cases to process. The input cases follow. The first line of each input case contains a single positive integer, n ($1 \leq n \leq 1{,}000{,}000$) representing the number of trampolines. Then, a single line follows with n integers, the values $| v_i | \leq 10^9$ for each trampoline.

The Output

For each case, output a single line with a single integer representing the answer to the input case.

Sample Input	Sample Output
2 7 1 -2 3 -1 2 -2 -2 5 4 2 5 -1 -3	5 0

Roads? Where We Are Going, We Don't Need Roads.

Filename: *roads*

Time Limit: *4 seconds*

The National Road Service (NRS) wanted to make some changes to make sure that people are correctly socially distancing from each other. However, NRS does not have the manpower to keep tabs on everyone. So, NRS wants to close down as many roads as possible while maintaining the connectivity between cities.

All of the roads in the existing road network are bi-directional roads connecting pairs of cities such that it is possible to travel between any pair of cities. NRS has taken a survey on roads, and come up with a priority ranking for each. The higher the priority score of a road, the more people can be on that road.

Normally, NRS would like to keep the higher priority roads. However, the NRS Director loves even numbers. This means that he wants to maximize the number of roads with an even priority score. Of all of these possible options, he wants to maximize the overall sum of priority scores of the roads selected.

The Problem

Given a network of roads, of all subsets of roads of minimum size that keep the whole network connected, determine the maximum sum of priority scores of the roads selected subject to the constraint of maximizing the number of roads chosen that have an even priority score.

The Input

The first line will have a positive integer, n ($n \leq 100$) with the number of road networks to be tested. The n test cases follow. The first line of each test case will have two integers, c ($2 \leq c \leq 1{,}000$) and r ($c\text{-}1 \leq r \leq 10{,}000$) representing the number of cities in the network and number of roads, respectively. The following r lines will each describe one of the roads. Each of these lines will contain 3 positive integers u, v, p ($1 \leq u, v \leq$ c, $u \neq v$, and $1 \leq p \leq 10{,}000$) where u and v represent the two cities connected by the road and p represents the priority points of the road. It is possible that more than one road will connect the same pair of cities.

The Output

For each input case, on a line by itself, output the total amount of priority points based on the rules the director has outlined.

Note: The Sample Input and Sample Output are on the following page.

Sample Input

```
2
3 3
1 2 4
2 3 8
3 1 12
5 7
1 2 10
1 3 9
3 2 2
4 5 7
5 1 2
2 5 17
3 5 4
```

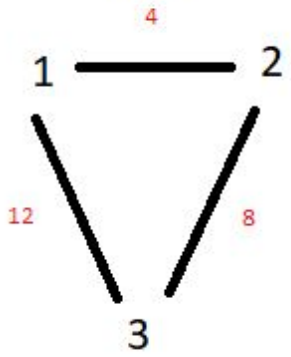
Sample Output

```
20
23
```

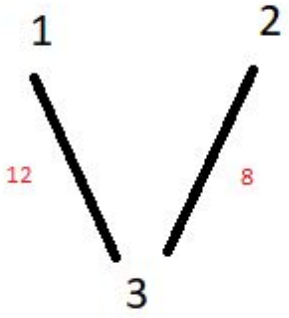
Note: The following page has an illustration of both sample cases.

Example 1

Before

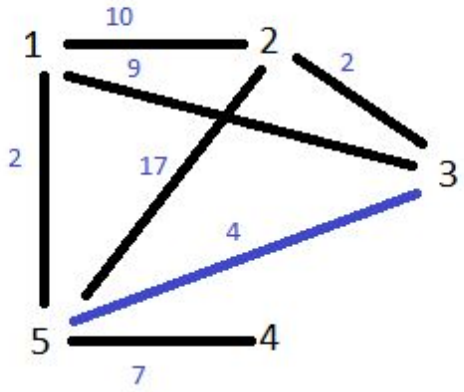


After

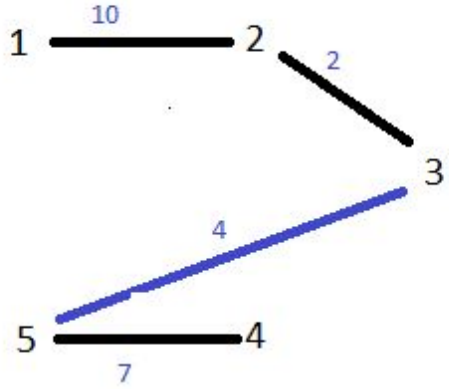


Example 2

Before

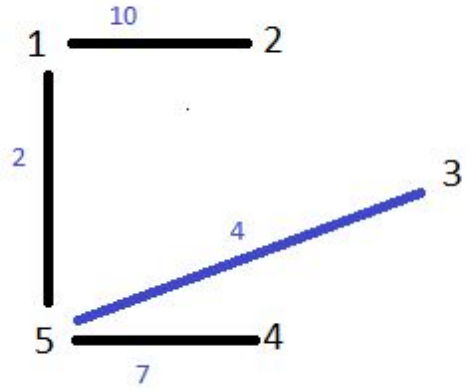


After



Or

The priority points are the same



Christmas Ornaments
Filename: *ornaments*
Time Limit: *3 seconds*

With fewer events to plan, the Orlando City Council has less work to do. So, they have decided to get an early start determining which ornaments will be on the Christmas tree this upcoming winter. Naturally, each council member has a favorite ornament, each of which are distinct, and furthermore, each council member wants the number of those ornaments hanging on the tree to be in some range, for example, in between 20 and 40, inclusive. Each tree that will be decorated in Orlando has a fixed number of ornaments that must be placed on it. With so many options, one of the council members wants to know how many ways there are to pick the quantity of each ornament while satisfying each council member's request while putting up the correct number of ornaments on the tree.

The Problem
Given the total number of ornaments that must go on a Christmas tree, along with each council member's request of how many of their favorite ornament they want on their tree, determine the number of ways the tree can be decorated while fulfilling all the requests of the council members. Since this number might be quite large, calculate its value modulo $10^9 + 7$.

The Input
The first line of input will contain a single positive integer, c ($c \leq 25$), representing the number of input cases to process. The input cases follow. The first line of each input case contains two space separated positive integers, n ($1 \leq n \leq 20$) and t ($1 \leq t \leq 1000$), representing the number of council members and the number of total ornaments to place on the tree, respectively. The following n lines will contain information about each council member's favorite ornament. Specifically, the i^{th} of these lines will have two space separated integers, L_i ($1 \leq L_i \leq 1000$) and H_i ($L_i \leq H_i \leq 1000$), representing that the i^{th} council member wants in between L_i and H_i, inclusive of her favorite ornament on the tree.

The Output
For each case, output a single line with a single integer representing the answer to the input case.

Sample Input

```
2
2 10
3 7
4 9
3 10
1 5
1 5
1 5
```

Sample Output

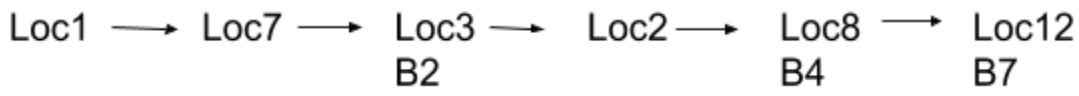
```
4
18
```

Scavenger Hunt
Filename: *scavenger*
Time Limit: *2 seconds*

You are participating in a virtual scavenger hunt, with *n* possible locations to visit. Whenever you visit a location, you are given a list of 0 or more choices of other locations to visit. If there are no choices, then your scavenger hunt ends, otherwise, you choose one of the given locations and proceed. Everyone starts at location number 1, and the choices are designed so that you can not visit any location twice, thus, it's guaranteed that your scavenger hunt will end.

In the old system, your score was simply the total number of locations visited. Unfortunately, a few competitors had figured out how to maximize their score under this system and the organizers wanted to make an exciting contest again, so they added a twist this year.

A few of the locations will be designated as bonus locations. If you visit a bonus location, then your score for the portion of the path starting at the bonus location until the end of the scavenger hunt is multiplied by the bonus multiplier assigned to that location, an integer in between 2 and 10, inclusive. Furthermore, bonuses can build upon each other! For example, consider the following path for a scavenger hunt:



Locations 3, 8 and 12, have bonus values of 2, 4 and 7, respectively. The last vertex contributes a value of 2*4*7 points because it benefits from the multipliers on locations 3, 8, and 12, while the first vertex only contributes 1 point because it isn't affected by any multipliers on the path. Therefore, the total value of the path above is {1 + 1 + 2+ 2 + 8 + 56} = 70.

While some contestants might want to maximize their score in the scavenger hunt, the director of the scavenger hunt has tasked you with answering a more interesting question: if a contestant always randomly selects which forwarding location to proceed to, what is their expected score for the scavenger hunt? In this way, the organizers can judge which contestants have done better than what they would have scored by chance. Help the organizers out!

The Problem
Given the layout of the scavenger hunt, including all of the forwarding locations, which locations are bonus locations and what each bonus location's bonus multiplier is, determine the expected score one would receive on the scavenger hunt if they randomly selected between all possible forwarding locations for each decision they made for where to go.

The Input

The first line of input will contain a single positive integer, c ($c \leq 100$), representing the number of input cases to process. The input cases follow. The first line of each input case has two space separated integers, n ($2 \leq n \leq 200$), representing the number of locations for the scavenger hunt, and b ($0 \leq b \leq n$), representing the number of bonus locations. The following b lines contain information about the bonus locations. On each of these lines there will be two space separated integers, v ($1 \leq v \leq n$), representing the location number and m ($2 \leq m \leq 10$), representing the multiplier bonus for that location. The locations that receive bonuses will all be unique.

The following n lines contain information about each of the locations, in order, from location 1 to location n. On the i^{th} of these lines, the first integer on the line, f_i ($0 \leq f_i < n$), represents the number of forwarding locations from location i. The following f_i values will be distinct integers in between 1 and n, inclusive, representing the locations to travel to on the scavenger hunt from location i. The input cases will be constructed such that no case has an answer greater than 10^9.

The Output

For each input case, output a single floating point number, rounded to two decimal places, indicating the expected score for a contestant who randomly selects between all possible forwarding locations for each movement decision. The data will be constructed such that adding or subtracting 10^{-6} from the actual answer won't change the answer rounded to two decimal places.

Sample Input

```
2
3 2
2 4
3 3
2 2 3
1 3
0
4 0
1 3
2 3 4
0
0
```

Sample Output

```
10.50
2.00
```

Essential Road Work
Filename: *essential*
Time Limit: *3 seconds*

There are n cities and $n-1$ open roads connecting them in such a way that there is exactly one path between any pair of cities. The king has hired m workers to work on the roads, but gave them very vague orders. The i^{th} worker must travel from city u_i to city v_i, and change the state of the road on his path from city u_i to city v_i. If a road is open on his path, the worker must close it. If a road is closed on his path, he must open it.

For example, if a worker's path goes from city 2 to city 5 to city 3 to city 7, and the road from city 2 to city 5 is open, the road from city 5 to city 3 is closed and the road from city 3 to city 7 is closed, after traveling his path, the road from city 2 to city 5 will be closed, the road from city 5 to city 3 will be open and the road from city 3 to city 7 will also be open. For the purposes of this problem, you may assume that each worker travels his path at a different time than any other worker.

The Problem
Given the configurations of roads between the cities, and a list of which pairs of cities each worker must travel between, determine which roads are open and which roads are closed after the workers complete their work on the roads.

The Input
The first line of input will contain a single positive integer, t ($t \leq 20$), representing the number of input cases to process. The input cases follow. The first line of each input case contains two positive integers, n and m ($2 \leq n \leq 100{,}000$), ($1 \leq m \leq 100{,}000$) representing the number of cities and the number of workers, respectively. Then, $n-1$ lines follow, each with two integers a and b ($1 \leq a, b \leq n$, $a \neq b$), meaning there is a road that connects cities a and b. Lastly, m lines follow, each with two integers, u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$), representing the start and end cities for every worker.

The Output
For each input case, print a single string on a line by itself of length n-1, where the i^{th} character is '1' if the i^{th} road listed in the input is open and the i^{th} character is '0' if the i^{th} road listed in the input is closed, after all the workers do their jobs.

Note: The Sample Input and Sample Output are on the following page.

Sample Input

```
2
4 3
1 2
1 3
1 4
2 1
1 4
2 3
2 3
1 2
1 2
2 1
1 2
```

Sample Output

```
100
0
```

Sample Explanation

In the first sample case, the road between cities 1 and 2 is closed by worker 1 and opened back up by worker 3. The road between cities 1 and 3 is closed by worker 3 and the road between cities 1 and 4 is closed by worker 2. The only road is closed, opened and closed in the second sample.