ACM-ICPC 2014
Asia Regional Contest
Kharagpur Site

IIT Kharagpur

December 7, 2014

You get:
10 Problems, 24 pages, 300 Minutes

This page intentionally left blank

Rules for ACM-ICPC 2014 Asia Regional Kharagpur Site

1. Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the results. Submitted codes should not contain team or University name and the file name should not have any white space.
2. The public rank list will not be updated in the last one hour. However, notification of accepted/rejected runs will **NOT** be suspended at the last one hour of the contest time.
3. A contestant may submit a clarification request to judges. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants.
4. Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee while seated at the team desk. **But they cannot even talk with their team members when they are walking around the contest floor to have food or any other purpose.** Systems support staff may advise contestants on system-related problems such as explaining system error messages.
5. While the contest is scheduled for a particular time length (five hours), the contest director has the authority to alter the length of the contest in the event of unforeseen difficulties. Should the contest duration be altered, every attempt will be made to notify contestants in a timely and uniform manner.
6. **A team may be disqualified by the Contest Director** for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior or communicating with other teams.
7. Team can bring up to 25 pages of printed materials with them. But they are not allowed to bring calculators or any machine-readable devices like mobile phone, CD, DVD, Pen-drive, IPOD, MP3/MP4 players etc. Blank papers will be provided, if needed.
8. With the help of the volunteers and external judges, the contestants can have printouts of their codes for debugging purposes. Passing of printed codes to other teams is strictly prohibited.
9. **The decision of the judges is final.**
10. **Teams should inform the volunteers if they do not get reply from the judges within 10 minutes of submission. Volunteers will inform the external judges and the external judge will take further action. Teams should also notify the volunteers if they cannot log in into the PC^2 system. This sort of complains will not be entertained after the contest.**
11. Ten problems will be posed. So far as possible, problems will avoid dependence on detailed knowledge of a particular applications area or particular contest language. All problems require you to **read test data from the standard input** as specified in the problem and **write results to the standard output**.

12. You can use any of the standard library functions that your chosen programming language provides. In addition, you can use the math library in C/C++. You cannot use any other library that requires an extra flag to be passed to the compiler command. If you do this, the judges will probably find a code "compilation error" in your program.
13. Your program is not permitted to invoke any external programs. For example, you cannot use in C the system call ("grep xyz abc") to determine whether the file abc contains an occurrence of the string xyz. *Violation of this rule may lead to disqualification from the contest.*
14. **Output must correspond exactly to the provided sample output format, including (mis)spelling and spacing.** Multiple spaces will not be used in any of the judges' output, except where explicitly stated.
15. Your solution to any problem should be submitted for judging using the PC^2 software only. Once you have submitted the solution, it will reach the judges. The time it takes for your problem to be judged will depend, among other things, on how busy the judges are. Once your submission has been judged, you will receive a message through PC^2 indicating the judgment. The judgment may be "Yes", meaning that your submission was judged to be correct. Otherwise you will get a message indicating the problem with your program. For example, the message may be "Wrong Answer", "Output Format Error", "Compilation Error", "Runtime Error", "Time Limit Exceeded" etc.
16. **Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required. The judges will only test whether the input-output behavior of your program is correct or not.**

Problem A: Drawing Contest

Input: Standard Input
Output: Standard Output
Problem Code: Drawing

Liliputs are holding a drawing competition for **K** kids, but with **K** human-sized pencils stolen from humans. In order to make life easier for the kids, the organizers want to give a pencil to each kid in a way such that the sum of the absolute differences of the height of a kid and the length of the pencil assigned to him/her is minimized. What is the minimum sum of absolute differences that can be achieved?

Input:

The first line contains the number of test cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains the number of kids and pencils **K** ($0 < \mathbf{K} \leq 100$). The second line contains **K** positive integers, each containing the height in millimeter of a kid. The third line contains **K** positive integers, each containing the length in millimeter of a pencil.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the minimum sum of absolute differences achieved (in millimeter).

Sample Input	Sample Output
2 5 51 60 51 72 70 65 60 88 72 88 4 123 234 188 175 219 334 122 233	Case 1: 69 Case 2: 190

This page intentionally left blank

Problem B: Sequence Containment

Input: Standard Input
Output: Standard Output
Problem Code: Sequence

A DNA sequence can be represented by a string of letters T, A, C, and G representing four different amino acids. DNA sequences are often matched to infer structural or functional similarities between living beings. Given two DNA sequences X and Y, the sequence Y is said to be contained in X if Y can be obtained from X by deleting 0 or more letters (not necessarily consecutive) in X.

Given two DNA sequences X and Y, what can be the minimum length of a third sequence Z such that both X and Y are contained in Z?

Input:

The first line contains the number of test cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains two integers **P** and **Q** ($0 < \mathbf{P}, \mathbf{Q} \leq 1000$) denoting the number of letters in the two sequences X and Y respectively. The second line contains the sequence X and the third line contains the sequence Y.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the minimum length of sequence Z.

Sample Input	Sample Output
2 7 6 TAGCTAG ATCATG 10 9 GGATGCTACA TCTACCGTA	Case 1: 9 Case 2: 13

This page intentionally left blank

Problem C: King's Dinner

Input: Standard Input
Output: Standard Output
Problem Code: Dinner

The King of Neverland plans to host a Christmas dinner for all the **K** landlords (numbered from 1 to **K**) in the kingdom. The landlords are feudal and the king feels it might be good to get them all together under one roof. However, he knows many of them really hate each other and seating them in the same dinner table may cause immediate problems. Luckily, he has two dinner tables in his banquet hall, and he sets out to see if he can sit all of them while making sure that two landlords who hate each other are not seated at the same table. He will hold the dinner only if this is possible. You need to help him decide whether he can invite the landlords or not given the above constraint. Also, if he can invite them, you need to help him find the number of people to be seated at each table. Note that the number of people to be seated at each table may not be unique, in which case, you need to find any one possible way. It is known that every landlord hates at least one other landlord, and if landlord *u* hates landlord *v*, then landlord *v* also hates landlord *u*.

Input:

The first line contains the number of test cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains the number of landlords **K** ($0 < \mathbf{K} \leq 500$). This is followed by **K** lines, where the j-th line contains the information about all the landlords that landlord j hates in the following format: the first integer **P** (> 0) in the line contains the number of landlords that landlord j hates, followed by **P** integers in the same line giving the id of the landlords that landlord j hates. Note that since landlord *u* hates landlord *v* implies that landlord *v* also hates landlord *u*, therefore if landlord *v* appears in landlord *u*'s list in the input, then landlord *u* appears in landlord *v*'s list in the input.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by 3 integers, separated by single space. The first integer is 0 if the landlords cannot be invited, and 1 if they can be invited. If the first integer is 0, the second and the third integers must both be 0. If the first integer is 1, then the second and the third integer show the number of landlords to be seated in the two tables, with the smaller number as the second integer and the larger as the third integer.

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

Problem D: Interview Scheduling

Input: Standard Input
Output: Standard Output
Problem Code: Interview

Professor Calculus wishes to hire graduate engineers for a few open positions in his funded projects. He has called **K** candidates for interview (numbered from 1 to **K**), and based on the profile of the candidates, has decided on an estimated time duration of interview for each candidate. A single interview board is going to conduct all the interviews. Assume that the interview board starts the interviews at time $t = 0$.

Since candidates from outside the town may want to go back on the same day, Professor Calculus has asked each candidate for the departure time of the train that they wish to take to go back after the interview. Many candidates have indicated such times, while others who are local candidates or non-local candidates who do not wish to go back the same day have not.

In order to cause any inconvenience to as few of the candidates as possible, Professor Calculus attempts to order the interviews such that the number of candidates who wish to go back the same day after the interview but will miss their trains is minimized. Assume that to catch a train, the interview of each candidate has to end at least 30 minutes before the departure time of the train the candidate wishes to take. You can also assume that all departure times specified are more than 30 minutes after the start of the first interview at $t = 0$.

Can you help the Professor Calculus prepare a schedule of the interviews so that the number of candidates who miss their trains is minimized?

Input:

The first line contains the number of test cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains the number of candidates **K** ($0 < \mathbf{K} \leq 100$). The second line contains a sequence of **K** positive integers indicating the estimated interview duration in minutes of the **K** candidates. The third line contains a sequence of **K** positive integers indicating the time, in number of minutes from $t = 0$, of the departure time of a train that a candidate has to catch. If a candidate has not given any such time, the corresponding integer is set to -1.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the minimum number of candidates who will miss their trains.

Sample Input	Sample Output
2 5 10 15 20 15 25 60 70 50 90 80 4 15 25 20 30 -1 75 70 70	Case 1: 2 Case 2: 1

Problem E: Robot Movement

Input: Standard Input
Output: Standard Output
Problem Code: Robot

Consider a **K×K** square made of $\mathbf{K}^2$ unit squares. The center of the lowermost, leftmost unit square has coordinates (0, 0) and the center of the rightmost, uppermost unit square has coordinates ((**K**-1), (**K**-1)). Consider a robot that can move only in a straight line and cannot change its direction once it starts moving. If the robot starts from the center (p,q) of a unit square to retrieve an object from the center (r,s) of another unit square, what is the number of unit squares it has to pass through? Note that touching any boundary point on a unit square also counts as passing through that unit square.

Input:

The first line contains the number of test cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains **K** ($1 < \mathbf{K} \leq 10000$), the dimension of the grid. The second line contains the x and y coordinates (in that order, separated by one or more spaces) of the starting point of the robot. The third line contains the x and y coordinates (in that order, separated by one or more spaces) of the destination point of the robot.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the number of unit squares passed.

Sample Input	Sample Output
3 8 0 7 3 0 8 1 0 7 1 8 3 3 7 7	Case 1: 12 Case 2: 8 Case 3: 13

This page intentionally left blank

Problem F: BlindDate Contest

Input: Standard Input
Output: Standard Output
Problem Code: BlindDate

Academia University is hosting BitFest at GridLand, with participants from Academia (host) and other colleges (guest). The first event is a team coding event BlindDate, with each team containing one student from host and one from guest. **H** host participants and **G** guest participants have assembled in GridLand. All of them wish to participate in BlindDate.

GridLand is shaped as a **K×K** grid, and each grid point in the grid is specified by a pair of coordinates. The lowermost, leftmost grid point has coordinates (0,0) and therefore the rightmost, uppermost grid point has coordinates (**K**, **K**). The grid points are connected by roads that run north-south and east-west along the edges of the grid, so that he distance between two grid points (*p*,*q*) and (*r*,*s*) is the Manhattan distance between them (= $|p-q| + |r-s|$). Each participant initially stands on a grid point (more than one participant can stand on the same grid point) and the game will finally be played in the Arena which is located at the grid point (**K**, **K**).

Professor BitGuy is organizing BlindDate and he will decide the teams. Once the teams are formed, the host member moves to the guest member, picks him or her up, and they both move towards the Arena. A participant can be a member of only one team. Every host participant moves with a unit speed towards the designated guest member of her team along the roads, and both of them then move with unit speed to the arena along the roads.

Prof. BitGuy has to form the teams at time $t = 0$ in a manner such that the competition can start at a given positive integer time **C**. Assume that the team formation takes no time. What is the maximum number of teams BitGuy can form?

Input:

The first line contains the number of test cases, **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains the integers **H** ($0 < \mathbf{H} \leq 250$), **G** ($0 < \mathbf{G} \leq 250$), **K** ($0 < \mathbf{K} \leq 1000$) and **C**. The next **H** lines contain the x and y coordinates (in that order, separated by one or more spaces) of the host participants. The next **G** lines contain the x and y coordinates (in that order, separated by one or more spaces) of the guest participants. All coordinates are integers.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the maximum number of teams that can be formed.

Sample Input	Sample Output
2 5 5 50 87 1 2 2 12 3 22 1 32 2 42 10 10 10 12 10 20 20 33 20 42 3 4 20 28 5 5 6 7 8 10 17 1 17 18 18 18 19 19	Case 1: 4 Case 2: 2

Problem G: Tour Planning

Input: Standard Input
Output: Standard Output
Problem Code: Tour

Professor PariBrajika wishes to plan a tour from Kharagpur to Himalpur using Pushpak Travels. There are **K** tourist locations serviced by Pushpak Travels, numbered from 1 to **K**, including Kharagpur (numbered 1) and Himalpur (numbered **K**).

It is only possible to travel directly from some locations to some other locations. For any pair of locations for which direct travel is possible, the travel from the originating city can start only at some specific time intervals of the day and the travel time depends on the starting time interval from the originating city because of different reasons such as vehicle type availability, traffic, and other environmental conditions. We will refer to each such 5-tuple <*u*,*v*,*s*,*f*,*t*> where *u* is the originating location, *v* is the destination location, *s* and *f* are the start and finish of a time interval within which travel can start at *u*, and *t* is the travel time in hours from *u* to *v* when starting within the closed interval [*s*,*f*], as a **route**. The elements *s* and *f* are specified as hours of the day, in 24-hour format (i.e. 0 to 23). For a particular origin-destination pair, you can assume that all intervals specified are non-overlapping. You can also assume that the routes are such that even with the above constraints, it is possible to reach Himalpur from Kharagpur passing through 0 or more of the other locations.

Professor PariBrajika cannot travel continuously without rest. The number of hours she can travel continuously without rest is equal to the "energy credit" she has. She can earn an energy credit of 1 hour for every hour of rest she takes (rest can be taken in integer multiples of hours only). However, she can rest only at any of the tourist locations. She spends an energy credit of 1 hour for every hour she travels. She may decide to rest any time irrespective of the energy credit she has (for example if no route is available for the current time or if a future route will enable her to reach Himalpur earlier through some other path). In that case, she has unspent energy credit left and can accumulate further energy credit. Energy credits can be accumulated cumulatively during the journey, subject to a maximum of 6 hours worth of travel time. Therefore, she cannot travel for more than 6 hours continuously in any case after the last recharge.

If Professor Paribrajika can leave Kharagpur anytime after midnight ($t \geq 0$), find the earliest time, expressed as the total number of hours elapsed since the start of the tour (i.e., the time she actually leaves Kharagpur), at which Professor PariBrajika can reach Himalpur. Assume that at the start of the journey she is fully rested and is good to go for 6 hours (i.e., has energy credits of 6 hours).

Input:

The first line contains the number of tests cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains the number of locations **K** ($0 < \mathbf{K} \leq 500$), and the total number of routes **M** ($0 < \mathbf{M} \leq 10000$). This is followed by **M** lines, with each line containing the information of a single route, specified by five integers *u*, *v*, *s*, *f*, *t* (in this order with one or more space between two successive integers). The integers correspond to the 5-tuple <*u*,*v*,*s*,*f*,*t*> as described earlier ($1 \leq u, v \leq \mathbf{K}$, $0 \leq s, f \leq 23$, $s < f$, $t > 0$).

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the earliest time in hours from the starting time to reach Himalpur.

Sample Input	Sample Output
2 3 3 1 2 0 2 3 2 3 3 6 6 2 3 7 12 6 5 10 1 2 0 2 5 1 2 3 23 6 2 3 0 4 3 2 3 5 8 1 2 3 9 23 2 3 4 0 10 5 3 4 11 14 6 3 4 15 23 5 4 5 0 2 2 4 5 3 23 6	Case 1: 12 Case 2: 26

Problem H: Pilgrimage

Input: Standard Input
Output: Standard Output
Problem Code: Pilgrimage

Professor PariBrajika is in Himalpur and wishes to go to Gyankut to attend a workshop. There are **K** temples on the way to Gyankut. The temples are numbered from 1 to **K**, Himalpur is numbered **0** and Gyankut is numbered **K+1**. She wishes to visit some of the temples on her onward journey from Himalpur to Gyankut, and the rest on her return journey from Gyankut to Himalpur. In her onward journey she must visit the chosen temples in ascending order of their numbers, and in her return journey she is required to visit the temples in descending order of their numbers. Any violation of this will waylay her and she will get permanently lost. For example, if **K** = 7, and she selects to visit (2,4,5) on her onward journey, she must travel from Himalpur to 2, 2 to 4, 4 to 5, 5 to Gyankut and then Gyankut to 7, 7 to 6, 6 to 3, 3 to 1, and 1 to Himalpur.

The estimated travel times between any pair of the locations (with no intermediate stops) are known. A trip from any location X to another location Y takes the same time as the trip from Y to X. Assume that Professor PariBrajika spends a fixed time **R** at each of the **K** temples and at Gyankut.

If she starts from Himalpur at time t = 0, what is the minimum estimated time in which Professor PariBrajika can complete the tour from Himalpur to Gyankut and back?

Input:

The first line contains the number of test cases, **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains the two integers, **K** ($0 < \mathbf{K} \leq 100$) and **R**. This is followed by **K+2** lines, with each line containing information about travel time from one location to all other locations. Each line starts with an integer id X of the location ($0 \leq X \leq$ **K+1**), followed by **K+1** pairs of integers (total 2**K+2** integers), with each pair containing the id Y ($0 \leq Y \leq$ **K+1**, $X \neq Y$) of another location and the estimated travel time (positive integer) to location Y from location X.

Output:

For each test case, in the first line, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating estimated minimum travel time.

Sample Input	Sample Output
2 3 10 0 1 100 2 200 3 300 4 400 1 0 100 2 100 3 200 4 300 2 0 200 1 100 3 100 4 200 3 0 300 1 200 2 100 4 100 4 0 400 1 300 2 200 3 100 5 10 0 1 63 2 126 3 189 4 252 5 315 6 378 1 0 63 2 51 3 102 4 153 5 204 6 255 2 0 126 1 51 3 116 4 232 5 348 6 464 3 0 189 1 102 2 116 4 71 5 142 6 213 4 0 252 1 153 2 232 3 71 5 132 6 264 5 0 315 1 204 2 348 3 142 4 132 6 63 6 0 378 1 255 2 464 3 213 4 264 5 63	Case 1: 840 Case 2: 886

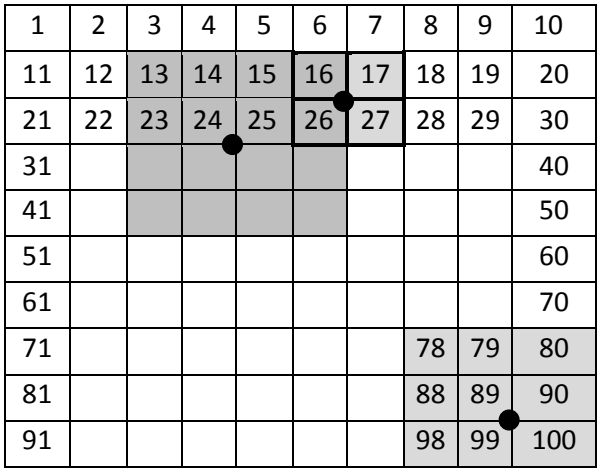
Problem I: Irrigating Farms

Input: Standard Input
Output: Standard Output
Problem Code: Irrigation

In the district of Jalshunyapur, the earth board has put restrictions on the usage of water in any farm to **C** litres per hour. Jalshunyapur has many farmers each having a farm of size **K** meter x **K** meter, divided into $\mathbf{K^2}$ square plots each having a side of 1 meter. The plots are numbered from 1 to $\mathbf{K^2}$ from left-to-right, top-to-bottom (see picture below).

To irrigate the plots, the farmer has put **S** sprinklers (numbered from 1 to **S**) at the bottom right corners of some of the plots. Each sprinkler, j, has a maximum possible flow, m[j] (positive integer); it can be programmed to work at any current integer flow between 0 and m[j]. If the current flow is set to c[j], then the sprinkler will be able to reach the adjacent plots up to a distance of c[j] on all sides. For example, for c[j] = 1, 4 plots will be watered; for c[j] = 2 and 3, the numbers will be 16 and 36 respectively. You can assume that a sprinkler can irrigate a plot if the flow of water from that sprinkler reaches the plot. Note that a sprinkler can be placed such that some of the plots it reaches fall outside the farm, in which case it irrigates only the plots falling inside the farm. In total, the sum of all c[j] can never be greater than **C**.

The picture below shows a 10×10 farm, with the plots numbered from 1 to 100 as shown. The sprinkler placed at the bottom right corner of plot 16 has c[j] =1 and irrigates 4 plots, and the sprinkler placed at the bottom right corner of plot 24 has c[j]=2 and irrigates 16 plots. Note that a plot may receive water from more than one sprinkler (plots 16 and 26 here). The sprinkler placed at the bottom of plot 89 has c[j]=2, but only irrigates the plots shaded as some of the plots it can reach are outside the farm.



The farmer plants different types of plants in each plot. Based on the type planted, each plot has a profitability value (positive integer). If a plot is irrigated, then the corresponding

profit is made. If a plot is not irrigated, then its profit earned is 0. There is no extra profit to the farmer if a plot receives water from more than one sprinkler.

What is the maximum profit the farmer can make by proper assignment of flows to the sprinklers?

Input:

The first line contains the number of tests cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, the first line contains three positive integers for **K** ($0 < \mathbf{K} \leq 10$), **S** ($0 < \mathbf{S} \leq 10$), and **C** ($0 < \mathbf{C} \leq 20$). The second line contains **S** integers separated by one or more spaces, with each integer representing the maximum flow **m[j]** of the corresponding sprinkler ($0 < \mathbf{m[j]} \leq 4$ for all j). The third line contains **S** integers, with each integer giving a plot number at whose bottom right corner the corresponding sprinkler is put. The fourth line contain $\mathbf{K^2}$ integers, with the j-th integer giving the profitability value of plot j.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer representing the maximum profit of the farmer.

Sample Input	Sample Output
2 3 2 2 2 2 3 7 1 1 9 1 1 1 8 1 5 4 2 2 2 2 2 12 1 1 1 1 1 1 1 1 1 1 1 1 1 8 1	Case 1: 19 Case 2: 13

Problem J: Finding Exponent

Input: Standard Input
Output: Standard Output
Problem Code: Exponent

Professor Calculus gives the following problem to his students: given two integers **X** (≥ 2) and **Y** (≥ 2), find the smallest positive integral exponent **E** such that the decimal expansion of $\mathbf{X}^{\mathbf{E}}$ begins with **Y**. For example, if **X** = 8 and **Y** = 51, then $\mathbf{X}^3$ = 512 begins with **Y** = 51, so **E** = 3. It is easy to see that neither **E** = 1 nor **E** = 2 will satisfy this property, so 3 is the smallest possible value of **E** in this case. Professor Calculus has also announced that he is only interested in values of **X** such that **X** is not a power of 10. The professor has a proof that in this case, at least one value of **E** exists for any **Y**.

The students set out to write a program for this, and run it on their department's server. However, they quickly find out that even though **X**, **Y**, and **E** will fit into a single integer, $\mathbf{X}^{\mathbf{E}}$ may be too large to fit into any single data type of any language they can program in. Can you help the students to find the value of the exponent **E**?

Input:

The first line contains the number of test cases **N** ($0 < \mathbf{N} \leq 3$).

For each test case, there is a single line containing the integers **X** and **Y**.

Output:

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer showing the value of the smallest exponent **E**.

Sample Input	Sample Output
2 5 156 16 4	Case 1: 6 Case 2: 3

END OF PROBLEMS