The 2009 ACM ASIA Programming Contest Dhaka Site

Sponsored by IBM

Hosted by North South University
Dhaka, Bangladesh

24th October, 2009
You get 19 Pages
11 Problems
&
300 Minutes

Updated Rules for ACM-ICPC 2009 Asia Regional Dhaka Site:

a) Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by a judge, and the team is notified of the results.

b) Notification of accepted runs will **NOT** be suspended at the last one hour of the contest time to keep the final results secret. Notification of rejected runs will also continue until the end of the contest. But the teams will not be given any balloon and the public rank list will not be updated in the last one hour.

c) 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.

d) Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee. Systems support staff may advise contestants on system-related problems such as explaining system error messages.

e) While the contest is scheduled for a particular time length (five hours), the Chief Judge 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.

f) **A team may be disqualified by the Chief Judge** for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, or distracting behavior. The external judges will report to the chief judge about distracting behavior of any team. **The external judges can also penalize a team with additional penalty minutes for their distracting behavior.**

g) Nine, ten or eleven problems will be posed. So far as possible, problems will avoid dependence on detailed knowledge of a particular applications area or particular contest language. Of these problems at least one will be solvable by first year computer science student, three will be solvable by a second year computer science student and rest will determine the winner.

h) Contestants will have foods available in their contest room during the contest. So they cannot leave the contest room during the contest without permission from the room invigilator. The contestants are not allowed to communicate with members of any other teams or coaches while they are outside the room. In other words rule (d) and (f) is valid within or outside the contest room.

i) Team can bring up to 200 pages of printed materials with them but they can also bring three additional books. But they are not allowed to bring calculators or any machine readable devices like CD, DVD, Pen-drive, IPOD, MP3/MP4 players, floppy disks etc.

j) With the help of the volunteers the contestants can have printouts of their codes for debugging purposes.

k) The decision of the judges is final.

l) Teams should inform the volunteers if they don't get reply from the judges within 10 minutes of submission. Volunteers will inform the External Judge 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.

m) If you want to assume that judge data is weaker than what is stated, then do it at your own risk :).

A That is Your Queue

Input: Standard Input
Output: Standard Output

Your government has finally solved the problem of universal health care! Now everyone, rich or poor, will finally have access to the same level of medical care. Hurrah!

There's one minor complication. All of the country's hospitals have been condensed down into one location, which can only take care of one person at a time. But don't worry! There is also a plan in place for a fair, efficient computerized system to determine who will be admitted. You are in charge of programming this system.

Every citizen in the nation will be assigned a **unique** number, from 1 to P (where P is the current population). They will be put into a queue, with 1 in front of 2, 2 in front of 3, and so on. The hospital will process patients one by one, in order, from this queue. Once a citizen has been admitted, they will immediately move from the front of the queue to the back.

Of course, sometimes emergencies arise; if you've just been run over by a steamroller, you can't wait for half the country to get a routine checkup before you can be treated! So, for these (hopefully rare) occasions, an expedite command can be given to move one person to the front of the queue. Everyone else's relative order will remain unchanged.

Given the sequence of processing and expediting commands, output the order in which citizens will be admitted to the hospital.

Input

Input consists of at most ten test cases. Each test case starts with a line containing P, the population of your country ($1 \le P \le 1000000000$), and C, the number of commands to process ($1 \le C \le 1000$).

The next C lines each contain a command of the form "N", indicating the next citizen is to be admitted, or "E x", indicating that citizen x is to be expedited to the front of the queue.

The last test case is followed by a line containing two zeros.

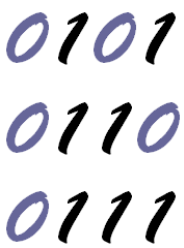
Output

For each test case print the serial of output. This is followed by one line of output for each "N" command, indicating which citizen should be processed next. Look at the output for sample input for details.

Sample Input	Output for Sample Input
3 6 N N E 1 N N N 10 2 N N 0 0	Case 1: 1 2 1 3 2 Case 2: 1 2

B How Many Ones Needed?

Input: Standard Input
Output: Standard Output



To write binary numbers we need only two digits '0' and '1'. To write a specific value we need a fixed number of ones, but of course number of zeroes may vary because of leading zeroes. For example to write values between 5 and 10 (inclusive) we need total 12 ones as shown in the figure on the left. You have to write a program that finds total number of ones that are required to write numbers in binary within a given range a and b.

Input

The input file can contain up to 11000 lines of inputs. Each line contains two positive integers a and b ($0 \le a \le b \le 2000000000$).

Input is terminated by a line containing two zeroes. This line should not be processed.

Output

For each line of input of input produce one line of output. This line contains the serial of output followed by an integer which denotes the number of '1' s required to write all the values between a and b (inclusive) in binary. Look at the output for sample input for details.

Sample Input	Output for Sample Input
5 10 20 30 0 0	Case 1: 12 Case 2: 35

C Foot Notes

Input: Standard Input
Output: Standard Output

I think most of you are familiar with the footnotes. They reside at the bottom of the page, describing various terms in that page. Some authors describe them as "Footnotes, the little dogs yapping at the heels of the text."

You are given an article of N lines, where some of which has keywords that need to be footnoted. Say, if you have the word "copotron"* in pages 1, 3, 7 and 8, then all these 4 pages should have a footnote about "copotron". Since, a page can have at most S lines, if you place M_i lines from the article in one page, you can place at most ($S-M_i$) numbers of footnotes at that page.

Each page will have a number of consecutive lines, and if, the last line of page i is line b, then first line of page (i+1) will be line (b+1).

Given the number of lines in the article, and the positions of the referenced words, find the minimum number of footnotes that has to be added.

Input

First line contains an integer **T (1 ≤ T ≤ 35)**, the number of test cases. This is followed by three integers **N (1 ≤ N ≤ 500)**, **S (1 ≤ S ≤ 100)** and **W (0 ≤ W ≤ 100)**, the number of lines in the article, number of lines that can fit in a page and the number of keywords. Next **W** lines each describe the positions of each keyword. Each of these **W** lines starts with an integer $\mathbf{K_i(1 \le K_i \le N)}$, the frequency of keyword **i**, followed by $\mathbf{K_i}$ integers, the lines in the article, where they can be found. The line numbers will be between 1 and **N**.

Output

For each test case, output the case number, followed by the minimum number of footnotes to add. If it's not possible to fit them, output -1.

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

**Copotron: As described in science fiction novel "Copotronic Shukh Dukhkho (Aka Emotions)" by M. Zafar Iqbal, Copotron is the brain of a robot.*

Description of the sample input/output

In case 1: if you place two lines in the first page, and 3 on the second, then, first page will require 1 foot note for first keyword since it contains keyword 1 in lines 1 and 2, and 2nd page will require 2 footnotes for keywords 2 and 3, since they appear in lines 3,4 and 5. You can achieve similar result, if you place lines 1-2 on first page, 3-4 on second, and 5 on third page.

In case 2: You can place at most 2 lines per page. There are 3 keywords, keyword 1 in lines 1 and 2, keyword 2 in lines 3 and 4, and keyword 3 in line 5. So, lines 1-5 each contain one keyword. We can fit at most two lines in a page. If we place 1 line in each page, then, we have to add 1 footnote as well. So, for 5 lines with keywords, we need 5 more footnotes. Please note that, it's not possible to place 2 lines in any page, since, it would require 2 footnotes, and thus a total of 2 + 2 = 4 lines, which is more than S.

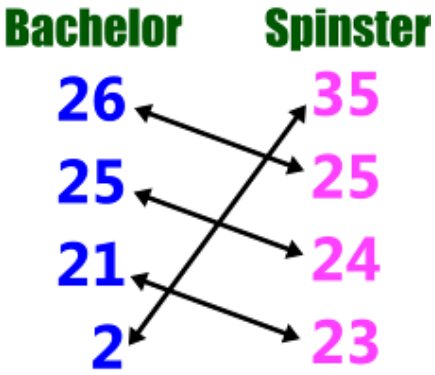
In case 3: You can not place line 1 in any page, if you place it anywhere, there wont be any space available to place the footnote, since maximum number of lines per page is 1.

D A Match Making Problem

Input: Standard Input
Output: Standard Output

Match-making is a tough job and even then its long term success (A happy family) depends on two people who are often not involved in the match making process. But now, sites like facebook, twitter and communication devices/software like mobile phones, messengers have made the professional match makers jobless. So these angry and jobless match makers have gathered together to make the government pass a law in the parliament that will stop people from choosing their life partners. The law is stated below:

In a certain community, the most senior bachelor must marry a spinster (Female Bachelor) whose age is nearest to him. Then next senior bachelor will then marry a spinster whose age is nearest to him (of course if there is a tie, marrying anyone of them will do) excluding the spinster that has already got married. This process continues until there is no bachelor or spinster left. Of course a bachelor cannot marry two spinsters and a spinster cannot marry two bachelors. For example in a community there are four bachelors who are 21, 25, 26, 2 years old and four spinsters who are 26, 24, 25 and 35 years old. The diagram below shows the only possibility of marriage: (eg: The 26 year old bachelor marries the 25 years old spinster)



Now given the ages of the bachelors and the spinsters in a community you will have to find the number of bachelors left, after all the marriages have taken place according to the law mentioned above. Also you have to report the age of the youngest bachelor left in the community if there is one.

Input

The input file contains at most 25 sets of inputs. The description of each set is given below:

The first line of each set contains two integers B (0<B<10000) and S (0<S<10000) which denotes the total number of bachelors and spinsters in the community respectively. Each of the next B lines contains one integer between 2 and 60 (inclusive)

which denotes the age of one bachelor in the community. Each of the next S lines contains one integer between 2 and 60 (inclusive) which denotes the age of one spinster in the community. For simplicity you don't need to worry about getting married at a very small age in this problem. That means unmarried people of all ages are valid bachelor or spinster.

Input is terminated by a line containing two zeroes.

Output

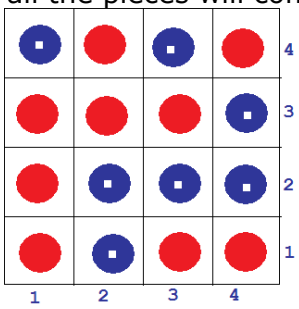
For each set of input produce one line of output. This line contains the serial of output followed by one or two integers. The first integer denotes the number of bachelors left in the community after all potential marriages have been completed. If this integer is not zero then print a second integer which denotes the age of the youngest bachelor left in the community after all possible marriages have been completed. Look at the output for sample input for details.

Sample Input	Output for Sample Input
4 4 26 25 2 21 35 25 23 24 1 2 20 30 40 4 2 5 5 10 15 20 18 0 0	Case 1: 0 Case 2: 0 Case 3: 2 5

E Game of Blocks

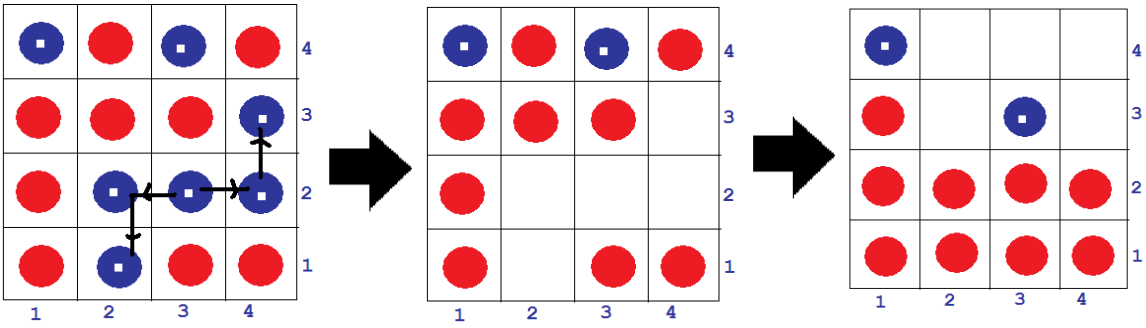
Input: Standard Input
Output: Standard Output

SOHA and TARA are playing *Game of Blocks*. 'Game of Blocks' is a two player game played on a 4x4 board. At the start of the game, each cell contains a piece colored red or blue. The two players make moves alternately – SOHA, being player 1, makes the first move. In each move, a player chooses a colored cell (that is a cell with a blue or a red piece in it). After making a move, all the reachable cells from that selected cell vanishes. Two cells are reachable if you can start from one and end on the other by making moves in the direction up, down, left or right and all the pieces in the path has the same color. After each move, all the pieces will come down to fill the empty spaces.



Consider an initial grid shown above. The blue pieces have white centers to differentiate them from the red pieces in order to facilitate the colored blind people.

Say, SOHA makes a move by selecting the cell at row 2 and column 3 or in other words cell (2, 3). The diagrams below shows what happens to the pieces after the move is made.



The player to remove the last piece wins the game. If both of the players play perfectly, can you determine whether SOHA will be able to win the game? If a player can win, (s)he will try to finish the game in minimum number of moves possible. However, if a player finds (s)he has no way to win, (s)he will make moves in order to delay the game as long as possible.

Input

The first line of input is an integer T(T≤1000) that indicates the number of test cases. Each case consists of 4 lines with 4 characters in each. The characters are 'B' or 'R' to indicate blue and red pieces respectively. There is a blank line after each case.

Output

For each case, output the case number first. If SOHA can win the game, print "win X", otherwise print "loss X" where X indicates the total number of moves before all the pieces vanishes.

Sample Input	Output for Sample Input
3 BBBB BBBB BBBB BBBB BBBB RRRR BBBB RRRR BRBR BRBR BRBR BRBR	Case 1: win 1 Case 2: win 3 Case 3: loss 4

F Password Remembering

Input: Standard Input
Output: Standard Output

Nadia wants to create two email accounts (one for her personal use and another for official use) in a popular emailing site named bestmail. One of the important parts of creating an account is choosing a password. Bestmail provides some rules for choosing a password. Here are the rules:

1. Password should be an integer number with no leading zero.
2. Password should not be less than a given integer A.
3. Password should not be greater than another given integer B.
4. Both A and B are provided by best mail as the range of password.

Nadia thinks choosing two different passwords for two different accounts is difficult to remember. So she decided she will choose one integer satisfying the rules for her first account and will use reverse of that integer for the second account. By this way she has to remember the password of first account only. Reverse of an integer can be found by reversing the order of the digits of the integer and removing the leading zeros. For example, reverse of 1203 is 3021, reverse of 1050 is 501.

After deciding this, she found that many integers are satisfying the rules but reverse of those are not always satisfying the rules. So she is interested about how many options she has to choose a password for her first account so that she can set the password for her second account just by reversing the first password.

Input

Input will start with an integer number T ($1 \le T \le 5000$), the number of test cases. Each test case contains two integers A and B ($0 < A \le B < 2^{64}$), range of password as described above.

Output

For each test case, output a single line of the form "Case X: N", where X denotes the case number and N denotes the number of options for Nadia for choosing a password for her first account.

Sample Input	Output for Sample Input
3 5 5 1 100 100 200	Case 1: 1 Case 2: 100 Case 3: 10

G Camera in the Museum

Input: Standard Input
Output: Standard Output

The Catenari Museum is world renowned for the valuable antique items it has. The total market value of all its items is around 20 trillion dollars and so guarding them is a very difficult job as antique buyers and businessmen are willing to pay any amount to get hold of them. So the Museum authority does not depend on human guards any more. They have placed many CCTV cameras in the Museum. Now they want to place one high powered, fast revolving CCTV camera at one position of the ceiling. As this type of camera is very costly, so they have decided to install only one such camera per floor.

All floors in the Museum are rectangular shaped and so it is possible to place the camera in a place from where all the valuables items on display can be seen. But there is a problem in the ground floor which also consists of the Head Office of the Museum. The head office has a circular shape and is somewhere strictly within the rectangular room. So it is possible that there are places in the floor from where all the valuable items are not visible. Your job is to find a place from where maximum number of items is visible and report this maximum number of valuable items. You can assume that the museum is so large compared to the valuable items that each valuable items can be denoted by a point in the 2 dimensional Cartesian coordinate system.

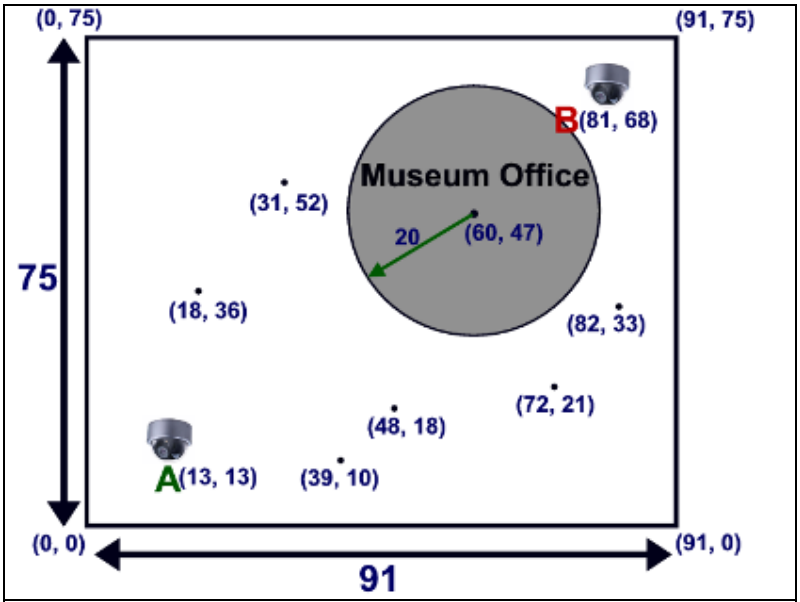


Figure: This denotes the first sample input. From location A six items are visible. But from location B less items are visible.

Input

The input file can contain up to 225 cases. But most of the cases are not extreme. The description of each test case is given below:

Each test case starts with six integers H, W ($50 \leq H, W \leq 10000$), R ($0 < 2R < 10000$), C_x, C_y, N ($N \leq 10000$). Here H and W denote the length and width of the room respectively, R denotes the radius of the circle that denotes Museum office, and (C_x, C_y) denotes the coordinate of the center of the circle. The circle is strictly within the rectangular room. You can also assume that walls of the museum are axis-parallel and the coordinate of the lower left corner of the room is at the origin (0, 0). So the coordinate of the upper right corner is (W, H). Each of the next N lines contains two integers which denote the coordinate of one valuable items of the museum. You can safely assume that no two items will be on the same coordinate, all items will be strictly inside the room but no items will be inside the Museum office or even on the boundary of office or room.

Input is terminated by a line containing six zeroes. The size of the input file is less than 5 Megabyte.

Output

For each line of input produce one line of output. This line contains the serial of output followed by an integer T which denotes the maximum number of valuable items that is visible from at least one point in the museum room. So if the coordinate of the point is (s, t) it must satisfy ($0 \leq s \leq W$ and $0 \leq t \leq W$). You don't need to print the coordinate of the place from where maximum number of items is visible because there may be more than one such place. Look at the output for sample input for details. You can assume that small precision errors will not affect the result. Use double-precision floating-point numbers for calculation.

Sample Input	Output for Sample Input
75 91 20 60 47 6 39 10 48 18 72 21 82 33 18 36 31 52 0 0 0 0 0 0	Case 1: 6

H Traffic Jam

Input: Standard Input
Output: Standard Output

Traffic Jam is becoming a major problem in the city of 'Faka' these days. Every day, you need to go to work from home through a one-way road that can be modeled as a series of straight line segments (aka piece-wise linear) of positive length. To be more explicit, suppose there are one way straight line paths named P, Q and R in your way. P starts at your home & ends at the starting point of Q. Q starts from the ending point of P & ends at starting point of R. Similarly, R starts at ending point of Q & ends at your destination. Two non-adjacent line segments in a road will never intersect with each other.

Now, you start from home for work and immediately get irritated by the heavy and annoying traffic jam. So you wish you could fly to your destination in order to avoid spending the whole day on road. Your new vehicle Aerocar comes as the solution to you. With this car, you can fly from any point on the road (The aerocar has a vertical take-off & landing similar to that of a helicopter) and land back to any point on the road any time. Now, though you would have liked to fly all the way and thus avoiding the traffic, you need to resist the temptation due to the fact that flying requires more fuel than driving. You need 1 unit of fuel to travel every unit distance along the path while F units per unit distance while flying. Now you need to write a program that, given the description of the road, can calculate the minimum possible fuel amount needed to complete your journey.

Input

Input file contains less than 60 test cases. Each test case starts with a couple of integers N ($1 \le N \le 25$) & F ($2 \le F \le 5$). N is the number of line segments in the road. F is the amount of fuel required for every unit distance traveled while flying. Next (N+1) lines each has two integers, x ($-1000 \le x \le 1000$) & y ($-1000 \le y \le 1000$). Assuming the road is placed on a planar 2d grid, the integers in the i'th line denotes the x and y co-ordinates of the starting point of the i'th line segment while the the end point co-ordinates by the i'th line segment are given at the (i+1)'th line. The last test case will be followed by a case with N = F = 0 indicating the end of input. This case should not be processed.

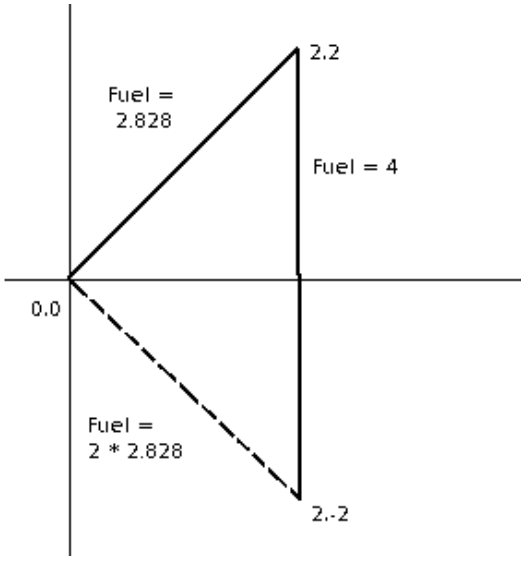
Output

For each test case except the last one, print one line of the form "Case X: Y", where X is the serial number of output & Y is the minimal possible units of fuel required. Print 3 digits after decimal point for Y. The output will be tested by a special judge program and outputs with precision error smaller than 10^{-3} shall be considered as correct.

Sample Input	Output for Sample Input
2 5 0 0 5 0 10 0 2 2 0 0 2 2 2 -2 3 5 0 0 1 1 -7 0 1 10 0 0	Case 1: 10.000 Case 2: 5.464 Case 3: 22.283

Illustration:

The following figure illustrates the second test case of sample input. Your home is placed in the (0,0) point whereas the destination is at (2,-2). If the travel is completed through the roads, the fuel consumption will be 6.828 units (2.828 units to go to (2,2) from (0,0) and 4 for the (2,2) to (2,-2) part). Alternatively, one can directly fly from (0,0) to (2,-2) with the required amount of fuel being 5.464 units (2.828 * 2). So, the best result is achieved by flying.



I Rectangle of Permutation

Input: Standard Input
Output: Standard Output

We want to build a rectangle where each row is a permutation of 0 to N-1. We want to make this rectangle with as many rows as possible while maintaining the following constraints.

$\sum_{j=0}^{N-1} E_{i,j} \le A_i$ and $\sum_{i=0}^{N-1} E_{i,j} \le B_j$, where $E_{i,j} = \begin{cases} (D_{i,j} - C_{i,j}) & when\ D_{i,j} > C_{i,j} \\ 0 & when\ D_{i,j} \le C_{i,j} \end{cases}$

$D_{i,j}$ is the number of occurrences of integer j in the column i. C is a matrix of N rows and N columns will be given as input. A and B are two sequences of size N will be given as input. Given N, A, B, C build a rectangle with the largest possible number of rows.

Input

First line of the input contains T ($1 \le T \le 50$) the number of test cases. It is followed by T test cases. Each test case starts with an integer N ($1 \le N \le 30$) indicating the number of columns in the rectangle. Next line contains N integers separated by single spaces. These integers are A_0 to A_{N-1} ($0 \le A_i \le 10$). Next line contains N integers separated by single spaces. These integers are B_0 to B_{N-1} ($0 \le B_i \le 10$). Each of the next N line contains N integers in each line. The integer on row i and column j is $C_{i,j}$ ($0 \le C_{i,j} \le 4$) (i and j starts from zero). A blank line will follow each test case.

Output

For each test case the first line of the output will be in the following format "Case C: R". Quotes are for clarity only. C is the test case number starting from 1. R is the maximum possible rows of the rectangle. Each of the next R lines should contain N integer in each line separated by spaces. Each of these N integers in each line should be a permutation of 0 to N-1. The whole RXN rectangle should maintain the constraints as described in the problem statement.

Sample Input

```
2
3
0 0 0
0 0 0
2 0 0
0 2 0
0 0 2

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

Output for Sample Input

```
Case 1: 2
0 1 2
0 1 2
Case 2: 7
0 1 2
1 0 2
1 0 2
2 1 0
2 1 0
2 1 0
0 2 1
```

J How Many Bases?

Input: Standard Input
Output: Standard Output

A classical problem of number theory is "Find the number of trailing zeroes in N^M, in base B". This problem is quite conventional and easy. But a number can have same number of trailing zeroes in more than one base. For example, if decimal number 24 is written in 3, 4, 6, 8, 12 and 24 based number system, it will look like 80, 60, 40, 30, 20 and 10 respectively. So in all number systems it has only one trailing zero. Given a number N^M, your job is to find out the number of integer bases in which it has exactly T trailing zeroes.

Input

The input file contains at most 10000 line of input. Each line contains three integers N ($1 \leq N \leq 10^8$), M ($0< M\leq 10^7$) and T ($0< T \leq 10^4$). The meaning of N, M and T is given in the problem statement. Input is terminated by a line containing three zeroes, which obviously should not be processed for calculation.

Output

For each line of input produce one line of output. This line contains the serial of output followed by an integer NB, which is modulo 100000007 value of number of bases in which N^M has exactly T trailing zeroes.

Sample Input	Output for Sample Input
24 1 1 100 200 10 23 18 2 0 0 0	Case 1: 6 Case 2: 312 Case 3: 3

K Integer Game

Input: Standard Input
Output: Standard Output

You are playing a single player game where you can convert one integer from another through a sequence of moves. The integer Y is reachable from X in a single move if the following is satisfied.

$Y = \frac{X \times P_2^k}{P_1^k}$, where k is a positive integer, P_1 and P_2 are prime numbers and X is divisible by P_1^k.

For example 18 is reachable from 8 in one move, because you can divide 8 by 4 and then multiply by 9. But 6 is not reachable from 8. Given two integers A and B calculate the minimum number of moves necessary to transform A into B. Both A and B can be very large. So each of them is needed to be expressed as a multiplication of a sequence of small integers: $A = \prod_{i=1}^{N} a_i$ and $B = \prod_{i=1}^{M} b_i$. Both of the sequences a_i and b_i will be given as inputs.

Input

First line of the input contains T ($1 \le T \le 40$) the number of test cases. Then T blocks of test cases follows. First line of the test case contains N ($1 \le N \le 300$) followed by N integers. N is the length of the sequence a_i and the following N integers form the sequence a_i. The second line starts with an integer M ($1 \le M \le 300$). M is the length of the sequence b_i and the following M integers form the sequence b_i. Each of integers in these two sequences will be between 2 and 200 (inclusive).

Output

For each case of input, print the serial of output followed by an integer: the minimum number of moves required to transform A to B. If it is impossible to transform A to B with any number of moves output -1 instead. If the minimum number of moves is greater than or equal to 20 print a -1 as well.

Sample Input	Output for Sample Input
4 1 4 1 9 2 2 2 2 3 3 1 8 1 6 2 32 11 3 27 25 13	Case 1: 1 Case 2: 1 Case 3: -1 Case 4: 3

Problem A – That's Your Queue by Derek Kisman (Analysis by Derek)

This is a fairly easy data structures problem. One way to solve it is to pretend the queue initially contains only the first 1,000 people. (You'll never see a higher-indexed person except from an expedite command.) Then just maintain the queue with linear-time updates.

Problem B – How Many Ones Needed by Shahriar Manzoor (Analysis by Sohel)
Problem: If you write down all the numbers from A to B in binary, how many 1s do you have to write?

A bruteforce solution that iterates over all the integers and counts the number of 1s will surely time out as the range of A and B can be as big as 2*10^9. There are around 10000 test cases and that calculates to a total complexity of O(10^4 * 2*10^9 * 32) = O(64 * 10^13). Even with fast computers we have these days, it will take around a month to produce the necessary outputs!

There are two approaches in solving this problem. One of them is dynamic programming but the code for this is a bit cumbersome. The easier approach is pattern recognition. The ith position consists of 2^i 0 bits followed by 2^i 1 bits that repeat eternally. Keeping this in mind, it's not difficult to find the total number of 1s in the range [1, X]. The required output could, thus, be found by calculating the result from [1, B] and then subtracting [1, A-1] from it.

Problem C – Foot Notes by Manzurur Rahman Khan (Analysis by Manzurur)

This is a dynamic programming problem. If you place line [i,i+1,i+2,...,j] in one page, then the number of lines to place will be (j-i+1). Count the number of different words in these lines. If that is X(i,j), then total number of lines needed for that page will be (j-i+1)+X(i,j) which should be less than or equal to S. So, if you are given line [i,i+1,i+2,....,N], then to find the minimum number of footnotes to place F(i), first place k lines in the first page, and find minimum number of footnotes for the rest of the lines:

F(i) = min {X(i,i+k-1) + F(i+k)} [1 <= i <= N,1<= i+k <= N,and k + X(i,j) <= S]

Problem D – A Match making Problem by Shahriar Manzoor (Analysis by Shahriar)

The problem basically asks you do find out the difference of two numbers and the minimum value of a collection of numbers. So scratch your head if you know how to find these two things (difference and minimum) but could not solve this problem.

Problem E – Game of Blocks by Sohel Hafiz (Analysis by Sohel)

Problem I – Rectangle of Permutation by Abdullah al Mahmud (Analysis by Abdullah)
The solution have two phase. In the first phase we need to calculate the largest R and in the second phase we need build the rectangle with R number of rows.
First phase, Let us build a flow network. For each value i from 0 to N-1 we will have four type of vertices.

$V_{i,0}$ is for the columns

$V_{i,1}$ is for the columns through which we will have excess flow than C

$V_{i,2}$ is for the integers through which we will have excess flow than C

$V_{i,3}$ is for the integers

We will have edges from source to $V_{i,0}$ and $V_{i,3}$ to sink but initially let us set these capacity to zero. Other edges in the graph is

For each i and j capacity from $V_{i,0}$ to $V_{j,3}$ is $C_{i,j}$

For each i capacity from $V_{i,0}$ to $V_{i,1}$ is A_i

For each i capacity from $V_{i,2}$ to $V_{i,3}$ is B_i

For each i and j capacity from $V_{i,1}$ to $V_{j,2}$ is infinity

Now we need to find largest capacity R from source to $V_{i,0}$ and $V_{i,3}$ to sink so that the total flow is R*N. Because of the constraints the upper bound of R is 150. We can increase R 1 by 1 until the total flow is less than R*N. Ok now we have found the largest number of rows in the Rectangle. The second stage is to build the rectangle. Let us build the matrix D as described in the problem statement.

$D_{i,j}$ is the number of integers j in the column i of the resulted Rectangle

So $D_{i,j}$ = flow($V_{i,0}$ to $V_{j,3}$) + flow($V_{i,1}$ to $V_{j,2}$)

The sum of the columns of the matrix is equal to R and the sum of the rows of the matrix is equal to R.
Now let us build a bipartite graph where each side contains N vertices.

The capacity from vertex i in the left side to vertex j in the right side i

Such a bipartite graph always have R disjoint perfect matchings and we need to find these R perfect matchings. You can find these perfect matchings by any standard matching algorithms.

Problem J – How Many Bases?
A cute Number Theory Problem which can be solved using inclusion-exclusion principle or other ideas.

Problem K– Integer Game by Abdullah al Mahmud (Analysis by Abdullah)

Let us consider the prime factorization of

This is a mid-level problem with a flavor of classical game theory!

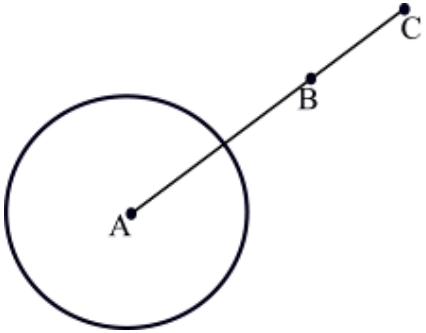
At any stage of the game, a particular cell can be empty or can contain any of the two colored pieces. Since the board size is only 4 x 4, there could be a total of 3^16 different states. But the nature of the game is such that a lot of those states will remain undiscovered and even map<string, int> to save the configurations should suffice.

Making moves: During a player's turn, try selecting each piece by bruteforce and use flood-fill to remove all the reachable pieces from it. Then do the necessary simulations to bring the pieces down until it can't descend any more.

Note: If you are not familiar on how game theory works, consult your 'Artificial Intelligence' book. :)

Problem F – Password Remembering by Arifuzzaman Arif (Analysis by Arif)

This problem can be solved using Dynamic programming(DP). Consider A and B as two string of length L (if string lengths are not equal, append some zeros in smaller one to make both equal). Try to append digits from left to right for the first password from position 1 to L, considering the restriction that it will remain within the lower bound A and upper bound B. When appending a digit after the prefix of the first password, you are actually also appending a digit before suffix of your second password. Make sure that after appending all digit your second password also remains in between A and B. So your DP state should be [length of prefix][length of suffix][state of prefix][state of suffix]. State of a suffix/prefix is whether it is already greater than A, already less than B, etc.



Problem G – Camera in a Museum by Shahriar Manzoor (Analysis by Shahriar)

Suppose a circular pillar is centered at A and you are at B and you can see m items. Now if you move further away from A along AB to point C and see n items. Then it can be shown that n>=m. So actually there is always a point on the wall of the room from where maximum number of items is visible. Now you have to implement it, need to know geometry ☺.

Problem H – Traffic Jam by Mohammad Mahmudur Rahman (Analysis by Mahmud)

Let us visualize the whole circumstance as a directed graph where interesting points along the road network are considered as nodes. The problem then reduces to finding the shortest path between the nodes at your starting location and the one at your destination. The endpoints of all the road segments are interesting points, as you might already have guessed. The tricky part is to observe that only a a few of the intermediate points from a road segment is going to be an interesting point. The following observation will help you - the angle of incidence from flight to road for a particular value of F is always fixed. You are encouraged to proof this idea. Once you have come to this, finding the interesting points requires some simple geometric calculation.

$$\frac{A}{B} = \prod P_i^{Q_i}$$

Now just forget about all the prime. Only consider the sequence Q(the power of all the prime factors). If the sum of this sequence is not zero then B is not reachable from A. Otherwise Q will contain some positive,negative and zero numbers. If Q contains all zero then A is equal to B and we are done. Otherwise let us consider the single move of transformation of X to Y. This is equavalent to adding k to one element of Q and subtracting k from another element of Q. In an optimal strategy we need to make at least one element of these two to zero. So a single move in optimal strategy is equavalent to removing a positive and a negative integer from Q and adding their sum to Q. The worst case number of minimum moves in such a strategy is size(Q)-1. But we can do better. Let us partition Q into maximum number of non-empty subsets where each subset sum is zero. Then the result is size(Q)-maximum number of non-empty zero sum partition of Q.
For sample test case 4

$$\frac{A}{B} = 2^3 \times 3^{-3} \times 5^{-2} \times 11^1 \times 13^{-1}$$
$$then\ Q = \{5, -3, -2, 1, -1\}$$

We can divide Q into 2 zero sum subsets. {5,-3,-2} and {1,-1}. So the result is 5-2 =3.

Now let us concentrate on partitioning the sequence Q into maximum number of non-empty zero sum subsets. A primitive pruning is to we can do is to greedily make pairs with sum zero. The rest of the task is to partition Q into non-empty subset with zero sum where the size of each of the subset is at least 3.An obvious solution is to execute a dynamic programming over all possible of subsets. This solution has a time complexity of $O(N2^N)$. N is the worst case size of Q and its upper bound is 46 (The number of primes between 2 and 200). So this solution is not feasible in such situation. One observation is that we don't need to visit all of these 2^N states . For example if we can determine the first subset in the optimal solution then its minimum size would be 3 then we need to consider another sequence with size less than or equal to N-3. This way the visited states in the optimal strategy will be small enough to be managable for memoisation. This leads to a recursive solution. In each recursize state let S be the set to be partitioned. Now let us generate the subset of S in the order of the length of those subsets. This can be done by a bfs over a bitmask. Once we find a subset S1 with sum zero let us calculate the partition for the set S-S1 recursively. Now we do not need to generate all the subsets in the bfs. Let the size of the current set in the bfs queue is l. Then the expected number of moves for this set is $|S| - \frac{|S|}{l}$. If this move is not less than the current best value then we do not need to visit the rest of the nodes in the bfs queue. The worst case number of nodes in the bfs queue is the choose($|S|, \frac{|S|}{2}$). The worst case value is choose(20,10). Because of the upper limit of output 19 the sets with size greater than 20 need to be partitioned into more than 2 sets so we need to break far before.