The 2011 ACM ASIA Programming Contest Dhaka Site

Sponsored by IBM

Hosted by North South University
Dhaka, Bangladesh

19th November 2011
You get 17 Pages
10 Problems
&
300 Minutes

Rules for ACM-ICPC 2011 Asia Regional Dhaka Site:

a) 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.

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 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.

e) 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.

f) **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 oe communicating with other teams. The **external judges** will report to the **Judging Director** about distracting behavior of any team. **The external judges can also recommend penalizing 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 two will be solvable by a first year computer science student, another one 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 external judges. **The contestants are not allowed to communicate with any contestant (Even contestants of his own team) or coach while are outside the contest floor.**

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 and external judges, the contestants can have printouts of their codes for debugging purposes. **Passing of printed codes to other teams is strictly prohibited.**

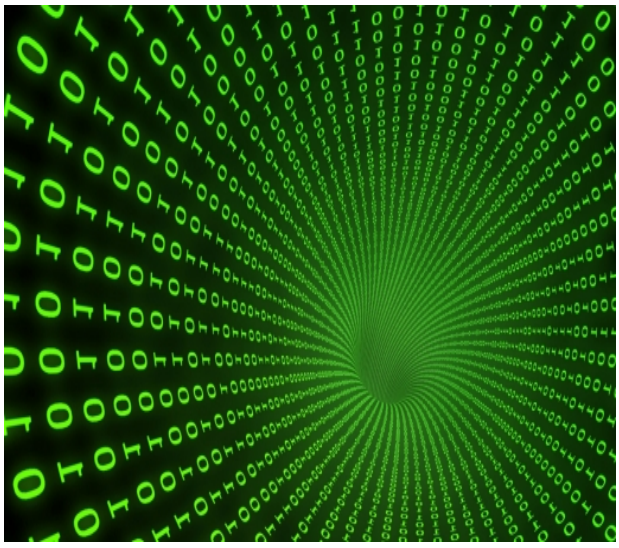
k) The decision of the judges is final.

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 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.

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

A Binary Matrix

Input: Standard Input
Output: Standard Output



You are given an **M × N** binary matrix. By **M × N** matrix we mean a matrix having **M** rows and **N** columns and by binary matrix we mean each of the **M × N** elements is a binary value, either **0** or **1**. In addition, this matrix wraps both in horizontally and vertically. So $\mathbf{i}^{th}$ row is adjacent to $\mathbf{(i + 1)}^{th}$ row for all $\mathbf{1 \le i < M}$ and $\mathbf{M}^{th}$ row is adjacent to $\mathbf{1}^{st}$ row. Similarly, $\mathbf{i}^{th}$ column is adjacent to $\mathbf{(i + 1)}^{th}$ column for all $\mathbf{1 \le i < N}$ and $\mathbf{N}^{th}$ column is adjacent to $\mathbf{1}^{st}$ column. Obviously row **a** is adjacent to row **b** implies that row **b** is adjacent to row **a**, and same thing is true for columns. Now, two cells of this matrix are adjacent if they are in the same row and their columns are adjacent, or they are in the same column and their rows are adjacent. So for a **3 × 5** matrix, cell **(2, 3)** has 4 adjacent cells **(1, 3), (2, 2), (2, 4), (3, 3)** and cell **(3, 5)** has **4** adjacent cells **(2, 5), (3, 4), (3, 1), (1, 5)**. Note that, by cell **(i, j)** we mean the cell of $\mathbf{i}^{th}$ row and $\mathbf{j}^{th}$ column.

You are only allowed to swap the values of any two adjacent cells of the matrix. Your task is to transform the matrix in such a way so that, each of the rows has same number of **1**s and each of the columns has same number of **1**s. If it is possible print **"both"** and also print the minimum number of swaps required. If it is not possible try to make every row has equal number of 1s. If it is possible print **"row"** and also print the minimum number of swaps required. If it is also not possible try to make every column has equal number of 1s. If it is possible print **"column"** and also print the minimum number of swaps required. If none of these possible you have to print **"impossible"**.

Input

The input starts with an integer **T (T ≤ 10)**, number of test cases.

Each case starts with two integers **M** and **N (2 ≤ M, N ≤ 1000)**, number of rows and columns of the matrix. Next **M** lines denotes **M** rows of the matrix. $\mathbf{j}^{th}$ character of the $\mathbf{i}^{th}$ line denotes the value of cell **(i, j)** of the matrix.

Output

For each case, output a single line. If task is impossible to complete, output **`"Case #: impossible"`** otherwise print **`"Case #: solution_type min_swap"`** without quotes, here # will be replaced by the case number, **solution_type** will be replaced by the type of solution found as described above it will be one of these three **"both"**, **"row"**, **"column"** without quotes and **min_swap** will be replaced by the minimum number of swaps required to complete the task. Please note that value of **min_swap** can be zero.

See the sample input and output for exact format.

Warning: Input file is large, so use fast input/output, for example instead of using cin/cout use scanf/printf.

Sample Input	Output for Sample Input
2 2 3 001 111 3 3 001 011 000	Case 1: row 1 Case 2: both 2

Explanation of sample input and output:

Case 1

The initial matrix is:

```
001
111
```

If we swap values of cell (1, 1) and cell (2, 1), the matrix will become

```
101
011
```

Now each row has two 1s and we found the solution.

Case 2

The initial matrix is:

```
001
011
000
```

If we swap the values of cell (2, 1) and cell (2, 3) (Considering the matrix wraps), the matrix will become

```
001
110
000
```

If we swap the values of cell (2, 1) and cell (3, 1), the matrix will become.

```
001
010
100
```

Now each row has one 1 and each column has one 1 and we got our solution.

B Candles

Input: Standard Input
Output: Standard Output

Figure 1: Digit-shaped candles

Figure 2: A cake with two candles denoting silver jubilee

It is the **68**th birthday of Animesh's father, so he is in "Archies's Gift Shop" of Banani (A place in Dhaka) to buy candles. He convinces his dumb head that buying 68 candles won't be such a good idea. But in the shop he finds some digit shaped candles and realizes that if he buys only two candles (One **'6'** and another **'8'**) he can display his father's age on the cake. But now he wants to buy digit shaped candles so that he can display ages of all the members of his family during their birthdays. Minimum how many candles does he need to buy? He has to obey the following restrictions:

1. There are only **10** pieces of candles in the store. **10** candles have the shape of **10** decimal digits as shown in figure **1**. Therefore he can buy at most one candle of each shape. There is another candle at Animesh's home which has the shape of **'+'**. So he does not need to buy that one. He can buy all ten digit shaped candles, but he wants to buy minimum number of candles, as for him these candles are not cheap.

2. He can only use the digit shaped candles to display an age, but also he can use the **'+'** shaped candle he already owns to display the age as sum of two numbers. For example if someone is **12** years old, he can display his age as **2+10**, **3+9**, **4+8**, **5+7**, **7+5**, **8+4**, **9+3**, **10+2** or **12**.

Input

The input file contains less than **10000** line of input. Each line starts with an integer **n (**$1 \le n \le 10$**)** which denotes total no of members in Animesh's family. This integer is followed by **n** integers, all of which are within the range **1** and **100** (inclusive). These integers denote the ages of the members of Animesh's family on their very next birthday. A line containing a single zero terminates input. This line should not be processed.

Output

For each line of input produce one line of output. This line contains the serial of output followed by some digits, which actually denotes the candles that Animesh must buy so that he can display the age of all his family members on their birthday. These digits are printed in descending order. These digits forms a number, lets call this number **T**. You can assume that their birthdays are in different dates so same candle can be used in more than one birthday. If there is more than one way to buy minimum number of candles, try to minimize the value of **T**. Look at the output for sample input for details. You can assume that there will always be a solution.

Sample Input	Output for Sample Input
2 10 11 1 30 0	Case 1: 654 Case 2: 30

C Cards

Input: Standard Input
Output: Standard Output

Taha has got a standard deck of cards with him. In addition to the 52 regular ones, there are 2 joker cards. Every regular card has a rank and a suit. The ranks in ascending order are: A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q and K. The suit of a card can be *clubs*, *diamonds*, *hearts* or *spades*. That means there are 13 *clubs*, 13 *diamonds*, 13 *hearts* and 13 *spades* - which adds up to 52. The joker cards have no ranks or suits.

One day, Sara gave Taha a challenge. First she randomly shuffles the 54 cards and starts placing one card after another, face-up, on a table. What is the expected number of cards Sara has to place so that there are at least **C** *clubs*, **D** *diamonds*, **H** *hearts* and **S** *spades* on the table? Whenever a joker card is encountered, Taha has to assign it to some suit so that the expected number of cards to reach the goal is minimized. The decision of assigning the joker card to some suit has to be made instantly (i.e. before Sara puts the next card on the table). Note that the assignments of the two joker cards don't necessarily need to be the same.

Input

First line of input is an integer **T**(T<50) that indicates the number of test cases. Each case consists of a line containing 4 integers in the order **C**, **D**, **H** and **S**. Each of these integers will be in the range [0, 15].

Output

For each case, output the case number first. Then output the expected number of cards Sara needs to place on the table to achieve the goal (rounded to 3 decimal places). If it's impossible to reach the goal, irrespective of what assignments Sara opts for, output "-1.000" (without the quotes) instead. Look at samples for exact format. Judge inputs are such that small precision errors won't cause error in the output. If the output is "0.000", make sure that you don't print it as "–0.000" (without the quotes).

Sample Input	Output for Sample Input
4 0 0 0 0 15 13 13 13 1 2 3 4 15 15 15 15	Case 1: 0.000 Case 2: 54.000 Case 3: 16.393 Case 4: -1.000

Illustration of the Samples:

1) There is no need to place any card as all required values are 0
2) We must place all the 54 cards to reach the goal
3) Note that output isn't always an integer
4) 60 Cards? No way!!

D Game of Connect

Input: Standard Input
Output: Standard Output

In the Game of connect two players are given a graph with n vertices and m edges. Before starting the game the first player selects two distinct vertices **A** and **B**. The goal of the second player is to connect **A** and **B** with a sequence of colored edges. Initially all the edges are uncolored. In his move first player can select an uncolored edge and deletes it from the graph. In his move second player can select an uncolored edge and colors it. If second player can connect **A** and **B** with a sequence of colored edges then he wins. Otherwise the first player wins. Players take their move by turns and the first player always goes first. Given a graph you need to determine whether second player has a winning strategy or not. Assume that both of the players play perfectly.

Input

First line of the input contains **T (T ≤ 100)** the number of test cases. Each test case starts with a line containing **n (2 ≤ n ≤ 100)** and **m (1 ≤ m ≤ 300)**. Each of the next m lines contains two integers **u** and **v** denoting an edge between **u** and **v**. The vertices are numbered from **0** to **n-1**. There will not be any duplicate edge.

Output

For each test case produce one line of output. This line contains the serial of output followed by a string **"YES"** if the second player has a winning strategy and **"NO"** otherwise (without the quotes). Look at the output for sample input for details.

Sample Input	Output for Sample Input
2 4 6 0 1 0 2 0 3 1 2 1 3 2 3 4 4 0 1 1 2 2 3 3 0	Case 1: YES Case 2: NO

E Guards

Input: Standard Input
Output: Standard Output

This ICPC will take place in a huge hall room which can be divided in to **N x N** square cells. That's why some volunteers will guard this room. But each row (or column) should be guarded by exactly two volunteers. And in a single cell at most one volunteer can be placed. Now volunteers can watch other volunteers either vertically or horizontally. Thus the volunteers form different groups. To be more specific, in a single group all the volunteers can look after each other directly or indirectly.

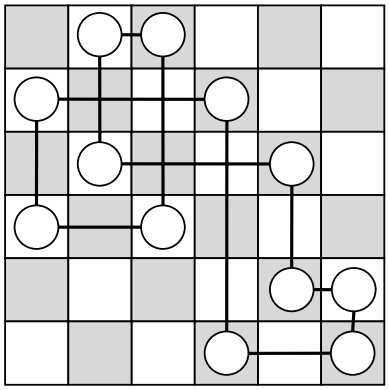


Fig 1

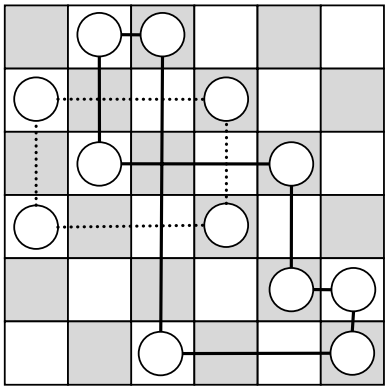


Fig 2

Suppose we have a hall room that can be divided into **6 x 6** square cells. Circles represent volunteers; lines represent the connectivity of the groups. In Fig 1, there is only one group (check the solid lines carefully). In Fig 2, there are two groups, one group is shown using solid lines, and another one is shown using dotted lines. Now the organizers wanted to know the number of ways they can place volunteers in the hall room such that they form exactly **K** groups. Two configurations will be different if in one configuration there is a volunteer on a cell but the cell is empty in another one. So, the organizers are seeking your help as you are one of the best programmers in town.

Input

Input starts with an integer **T (≤ 50000)**, denoting the number of test cases.

Each case starts with a line containing two integers: **N** and **K ($2 \le N \le 10^5$, $1 \le K \le \min(N, 50)$)**.

Output

For each case, print the case number and the number of ways the volunteers can be placed in the hall room as guards. The result can be large, so print the result modulo **1000 000 007**.

Sample Input	Output for Sample Input
4 2 1 3 1 4 1 4 2	Case 1: 1 Case 2: 6 Case 3: 72 Case 4: 18

F Packing for Holiday

Input: Standard Input
Output: Standard Output

Mr. Bean used to have a lot of problems packing his suitcase for holiday. So he is very careful for this coming holiday. He is more serious this time because he is going to meet his fiancée and he is also keeping frequent communication with you as a programmer friend to have suggestions. He gets confused when he buys a gift box for his fiancée because he can't decide whether it will fit in his suitcase or not. Sometimes a box doesn't fit in his suitcase in one orientation and after rotating the box to a different orientation it fits in the suitcase. This type of behavior makes him puzzled.

So to make things much simpler he bought another suitcase having same length, width and height, which is 20 inches. This measurement is taken from inside of the box. So a box which has length, width and height of 20 inches will just fit in this suitcase. He also decided to buy only rectangular shaped boxes and keep a measuring tape in his pocket. Whenever he chooses one gift box, which must be rectangular shaped, he quickly measures the length, width and height of the box. But still he can't decide whether it will fit in his suitcase or not. Now he needs your help. Please write a program for him which calculates whether a rectangular box fits in his suitcase or not provided the length, width and height of the box. Note that, sides of the box must be parallel to the sides of the suitcase.

Input

Input starts with an integer **`T (T ≤ 100)`**, which indicates the number of test cases.

Each of the next T line contains three integers **`L`**, **`W`** and **`H (1 ≤ L, W, H ≤ 50)`** denoting the length, width and height of a rectangular shaped box.

Output

For each test case, output a single line. If the box fits in the suitcase in any orientation having the sides of the box is parallel to the sides of the suitcase, this line will be **`"Case #: good"`**, otherwise it will be **`"Case #: bad"`**. In your output **`#`** will be replaced by the case number.

Please see the sample input and sample output for exact format.

Sample Input	Output for Sample Input
2 20 20 20 1 2 21	Case 1: good Case 2: bad

G Pair of Touching Circle

Input: Standard Input
Output: Standard Output

Given a rectangular grid of height **H** and width **W**. A problem setter wants to draw a pair of circles inside the rectangle so that they touch each other but do not share common area and both the circles are completely inside the rectangle. As the problem setter does not like precision he also wants their centers on integer coordinates and their radii to be positive integers as well. How many different ways can he draw such pair of circles? Two drawings are different from each other if any of the circles has different center location or radius.

Input

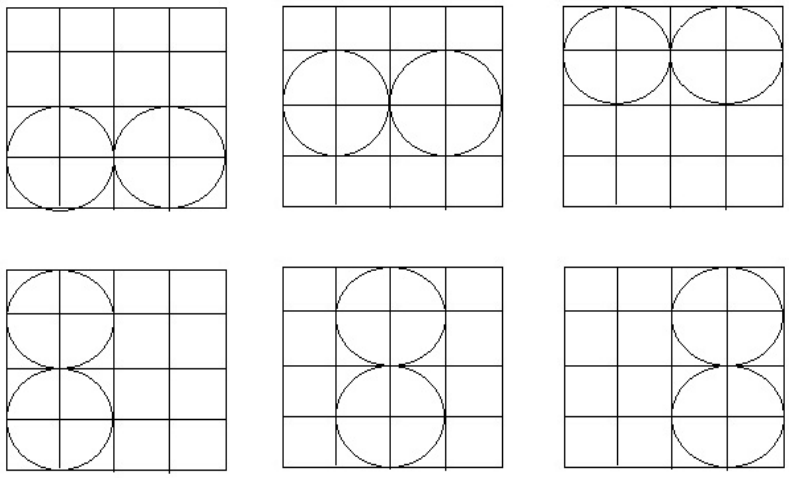
The first line of input contains the number of test cases **T** (**T** ≤ **500**). Each of the next **T** lines will contain two integers **H** and **W** (**0** < **H, W** ≤ **1000**).

Output

For each line of input output the case number and the number of ways of drawing such pairs of circles maintaining the mentioned constraints. See sample output for exact formatting. The output will fit into **64**-bit signed integer.

Sample Input	Output for Sample Input
5 4 2 4 3 4 4 4 6 10 10	Case 1: 1 Case 2: 2 Case 3: 6 Case 4: 16 Case 5: 496

Illustration of case 3:



H Treasure Hunt

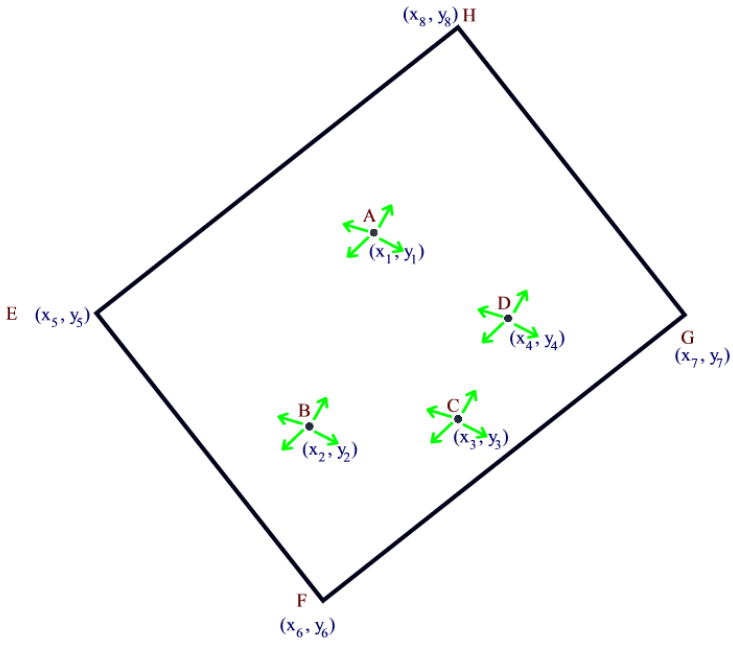
Input: Standard Input
Output: Standard Output

A plane of equal thickness is placed on a pillar and four persons jump on it.

They run to change their position and keep the plane balanced.

You have to help a group courageous people to find the lost treasure of your country, like the movie "**National Treasure** - *Book of Secrets*." To discover the national treasure that courageous group has to pass through many traps, solve many riddles and so on.

Like the film there are four people in the group and to reach the treasure they have to jump on a plate that is placed on a pillar with sharp edge at the top. Before any of them jumps on the plate, it is in a stable position. For simplicity you can assume that the plate is rectangular in shape, has equal thickness at all places, it is solid and made of same material. Four people of equal weight jump on the plate at the same time but they can jump only at some certain points of the plate. So they cannot always jump in such a way that the plate is stable after they jump on it. Therefore after jumping on the plate they must run and change their position to make the plate stable again. Given the shape of the plate (always rectangular) and location of the four people just after jumping on it, your job is to find the minimum distance they have to cover in total to make the plate stable again. After jumping the four people can run towards any direction. You don't need to print the minimum distance, but you need to print the final position of the four people after covering the minimum distance in total.



After jumping on the rectangular plate the four people can run at any direction they want.

Input

The input file contains around **15000** sets of inputs. Each set is described with **16** floating-point numbers scattered in two lines. The first line contains eight floating-point numbers $x_1, y_1, x_2, y_2, x_3, y_3, x_4$ and y_4 **(-10 ≤** $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$ **≤ 1010)** that denotes the coordinates of the four people just after jumping on the plate. So the coordinate of the four people are (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) respectively. The second line contains another eight floating-point numbers $x_5, y_5, x_6, y_6, x_7, y_7, x_8, y_8$ **(-10 ≤** $x_5, y_5, x_6, y_6, x_7, y_7, x_4$ **≤ 1010).** These eight floating-point numbers actually denote the coordinates of the four corners of the plate in counter clockwise order. The shape of the plate is always rectangular. You can assume that the jumping locations (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) are strictly within the plate boundary and all four people have equal weight. Input is terminated by two lines, each containing eight zeroes. All floating-point numbers in the input has twelve digits after the decimal point. The coordinates are all in two dimensions so they actually are given to specify the size and orientation of the plate and the location of the people with respect to the plate. So if a person stands still on the plate and the plate moves up and down or rotates, the coordinates of the person or coordinate of four corners of the plate does not change. In the sample input it is shown that each line contains four floating-point numbers. But it is only for lack of horizontal space on paper. In the judge input file there are eight floating-point numbers in each line.

Output

For each set of input produce four lines of output. Each line contains two floating-point numbers that denote the final position of one of the four persons. The final position of the four persons should be mentioned according to the same order given in the input and should make the plate stable. Also the final position of all four persons must be inside (or on the boundary of) the plate as well and ensure that the total movement made by the four persons is minimum. If there is more than one solution, any one will do. All floating-point numbers in the output should contain **12** digits after the decimal point. Print a blank line after output for each set. There is an special judge for this program, so small precision error will be ignored.

Warning: Input file is large, so use fast input/output, for example instead of using cin/cout use scanf/printf.

Sample Input

```
731.637000000000 437.595000000000 296.162000000000 402.836000000000
493.625000000000 260.917000000000 526.376000000000 237.611000000000
631.933553096878 841.348627667446 158.076619219714 651.913864882162
360.066446903122 146.651372332554 833.923380780286 336.086135117837
0.000000000000 0.000000000000 0.000000000000 0.000000000000
0.000000000000 0.000000000000 0.000000000000 0.000000000000
0.000000000000 0.000000000000 0.000000000000 0.000000000000
0.000000000000 0.000000000000 0.000000000000 0.000000000000
```

Output for Sample Input

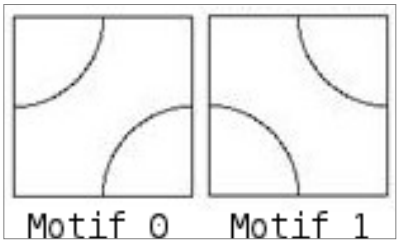
```
715.687000000000 596.855250000000
280.212000000000 562.096250000000
477.675000000000 420.177250000000
510.426000000000 396.871250000000

```

I Truchet Tiling

Input: Standard Input
Output: Standard Output

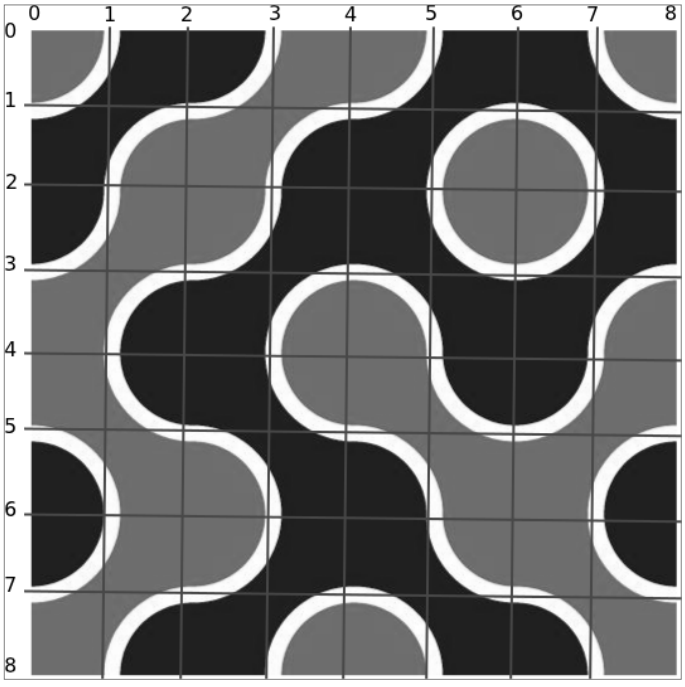
In 1704 French mathematician Sebastien Truchet proposed a tiling system that started with simple motifs and used it's rotations and random placement to create quite interesting tiling patterns. In this problem we will calculate the area enclosed by the curves in a special case of his tiling system.



Observe the two motifs here. Motif 0 is a square of length 2 units on each side. There are two circles with radius 1 unit drawn at two opposite corners of the square. Motif 1 is just a 90 degrees rotation of the first one (or you can think of it as drawing the circles on the other two corners).

Using just two basic motifs as shown here, we can tile an area to create rather interesting artistic patterns. They do not have to be symmetrically placed – we can lay out these motifs randomly to cover an area. If we then use a tool like paint bucket (found in most paint programs) at a particular point in the pattern, we can color a contiguous region with one color.

In this problem you'll be given a description of such a tiling pattern and then be asked how much area would it color if we use paint bucket starting at specific points.



Here is one illustration:

We have the motifs randomly placed on a **9x9** grid. Then we used a black color at position **(2, 4)** (Here **2** denotes the distance from the top and **4** denotes the distance from left). It colored all the contiguous region it found bounded by the grid's boundary and the curves themselves. We could achieve the same effect if we used the color at position **(4, 6)**, **(0, 6)**, **(1, 7)** etc. However, using the paint bucket on the perimeter of the curves (such as **(3, 4)**) will only color the perimeter line of the curves. They will not fill up a region with any color.

Input

The first line of the input starts with the number of test cases **T (1 ≤ T ≤ 100)**. In the next **T** lines you get the description of **T** test cases. For each test case the first line of the

test case gives the dimension of the area in rows and columns format. You can assume that rows **R** and columns **C** are at most **100** units long **(0 < R, C ≤ 100)**. For each of the next **R** lines you'll have the row description given in binary form. The basic motifs are numbered **0** and **1**. For example, the accompanying picture's first row can be described as **"0001"**. After the **R** row descriptions you'll be given **Q (1 ≤ Q ≤ 100)** – the number of queries. Each query will be of the form **x (0 ≤ x ≤ 2R)** and **y (0 ≤ y ≤ 2C)** where **x** and **y** are integers denoting the row and column number where you will use the paint bucket tool.

Output

For each test case print the test case number as **"Case C:"** in one line where **C** is the test case number. That line will be followed by **Q** lines of output for the queries on that test case. For each query print one number giving the area enclosed by the boundary and the curves that contain that point. If the point in question itself lies on a curve, you can assume the enclosed area to be zero. Your answer for each query must be rounded to **4** (four) digits after the decimal point. If the output is **0** (zero) make sure that you don't print it as **"-0.0000"**. Other than that inputs will be such that small precision error will not cause difference in output.

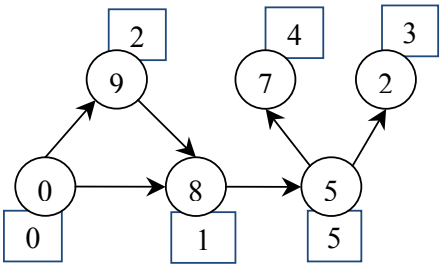
Sample Input	Output for Sample Input
3 1 2 01 4 0 0 2 0 0 1 0 2 2 2 01 00 1 2 2 3 1 1 0 1 2 3 1 4 2	Case 1: 0.7854 4.8584 0.0000 4.8584 Case 2: 4.7854 Case 3: 7.2876 1.5708

J As Long as I Learn, I Live

Input: Standard Input
Output: Standard Output

What makes problem solving so interesting? What makes it challenging? Why we put so much effort on it but never get bored? For us (problem-setters), it's like 'As long as I learn, I live!' We learn so many beautiful things, we find great ideas and techniques and there is no end to it.

In this problem, you are given a person's life stages in graph form. There are **n** nodes in the graph. Nodes represent stages, numbered from **0** to **n-1**, and edges represents that he can move from one stage to another. **0th** node is the node where he starts his journey. You may have guessed; there will be no cycles in the graph (no one can back to past!).



In each node, a value **x** is attached, it means he will learn **x** units, if he comes into this node. For example, the graph in the picture represents a person's life stages. The circles present nodes, the value in a circle represents that it will be gained if the person comes into this node. The numbers in rectangular boxes represents the node ids. If the person reaches the 1st node he will learn 8 added units, if he reaches 4th node, he will learn 7 added units.

As in real life, no one knows the future, but can predict some common things in near future. If the person is in node **u**, he only knows the nodes that have an **edge** from **u**. The person wants to learn more, that's why, he always takes the stage that looks better, i.e. has maximum value. So, if he is in node 0, he only knows the learning units in node 1 and 2, but doesn't know about nodes 3, 4, or 5. He will prefer node 2 over node 1 (because node 2 will give him 9 learning units). He continues journey in this method. He can finish at any stage, but will try to learn more. You can assume that the graph is given such that from any node, the next node can be picked deterministically i.e. from any node **u**, there will be **exactly one** node **v** which has the maximum value and there is an edge from **u** to **v**.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n (2 ≤ n ≤ 100)** and **m (n-1 ≤ m ≤ n*(n-1)/2)**, **n** denotes the number of nodes in the graph and **m** denotes the number of edges in the graph. The next line contains **n** space separated integers denoting the learning units in the nodes respectively. The values will be between **1** and **1000** (inclusive) except the **0th** node will have a value 0. Each of the next **m** lines contains two integers **u v (0 ≤ u, v < n, u ≠ v)** meaning that there is a directed edge from **u** to **v**. You can safely assume that the given graph will follow the restrictions described above. And from **0th** node, it can be possible to go to any node. There will be at most one edge between any pair of nodes.

Output

For each case, print the case number and the maximum total learning units the person can gain (following the strategy mentioned above) and the node id where the person ends journey.

Sample Input	Output for Sample Input
1 6 6 0 8 9 2 7 5 5 4 5 3 1 5 0 1 0 2 2 1	Case 1: 29 4