The 2010 ACM ASIA Programming Contest Kuala Lumpur Site

Sponsored by IBM

Hosted by

International Islamic University Malaysia

9th December, 2010
You get 18 Pages
10 Problems
300 Minutes

Rules for ACM-ICPC 2010 Asia Regional Kuala Lumpur Site:

a) Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judges, 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 Regional Contest Director in consultation with 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** for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, or 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 volunteers. 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) With the help of the volunteers, the contestants can have printouts of their codes for debugging purposes.

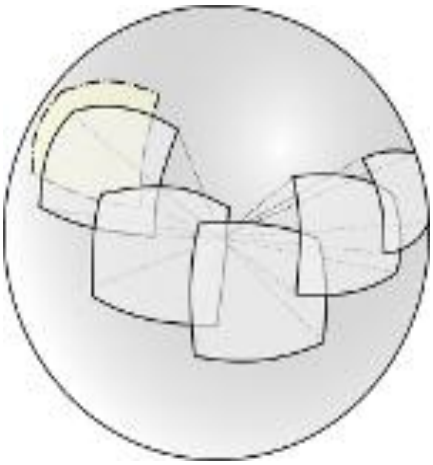
j) The decision of the judges is final.

k) Teams should inform the volunteers if they don't get reply from the judges within 10 minutes of submission. Volunteers will inform the Chief Judges and the Chief 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.

l) If you want to assume that judge data is weaker than what is stated in the problem statement, then do it at your own risk.

A Overlapping Scenes

Input: Standard Input
Output: Standard Output



The Dummywood film industry is infamous for its repetitive stories. The phrase, "if you have seen one, you have seen them all" perfectly applies to it. The movies are made by fitting together existing set of scenes. This reduces production time, as there is hardly ever a new stunt to perform, nor the writers need to be creative in bringing out something innovative. To make the movies lengthy, the producers opt to repeat monotonous events, such as repetitive breaking of glass doors or the "bad guys" rolling over the floor. Cutpiece is a producer who wants to use the power of computers to gain advantage in this film industry. He has a set of scenes out of which he wants to make his next movie. He plans to merge these scenes to make one complete movie. Although, mere merging of the scenes in some order would make a movie, but Cutpiece wants to minimize the length of the movie. The minimizing technique that he plans to apply relies on the fact that, scenes are so repetitive in their components that, the beginning of one and ending of another may be identical up to a certain length. Cutpiece has decided to condense these scenes so that the repetitive portions are included once in the merging process. In this problem, given a set of scenes, you will have to determine the minimum length of the movie that can be made by merging the given scenes in a particular order, condensing the repetitive portions during the merging process. Look at the explanation for sample input/output below to further clarify the condensing process.

Input

The first line of input will consist of a positive integer **T ≤ 50**, where **T** denotes the number of test cases. Each case starts with a positive integer **n ≤ 6** where **n** denotes the number of scenes that Cutpiece will merge. The following **n** lines will each contain a string of length at most 10. The strings will consist of upper case letters only and will have at least one character. Each of these strings represents one scene and the individual letters correspond to components forming a scene.

Output

For each case of input, there will be one line of output. It will consist of the case number followed by the minimum length of movie that can be made. Look at the output for sample input for exact formatting.

Sample Input	Output for Sample Input
2 3 ABCD DEFGH CDEF 2 AAAAA AAAAAAA	Case 1: 8 Case 2: 7

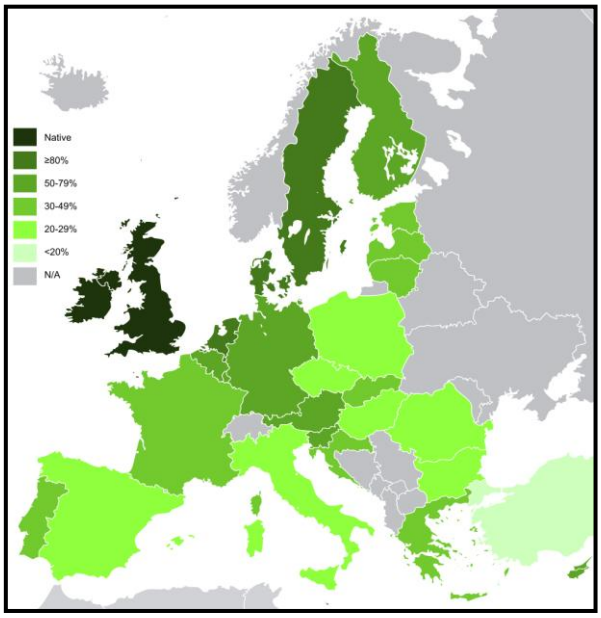
Explanation of Sample Input/Output

Case 1 -> if we order the input strings as "ABCD" "CDEF" and "DEFGH". Merge the first 2 to get "ABCDEF". Here we condense (CD) into a single occurrence since this is the longest length common suffix of one and prefix of another. Next we merge ABCDEF with DEFGH to obtain ABCDEFGH, giving us a string of length 8. Any other ordering of the three strings will not yield a shorter final string. Note that, when merging, we always merge from left to right after ordering the string. Therefore, for the above ordering, we would not merge CDEF and DEFGH first.

Case 2-> Here one string is a subset of another and we can entirely condense the components of shorter string to give us a string of length 7.

B Language Detection

Input: Standard Input
Output: Standard Output



English, Spanish, German, French, Italian and Russian are the 6 most prominent languages in the countries of European Union. Figure on the left shows intensity of English speaking people in different European countries. All of these languages have different words to represent the English word "HELLO". For example in Spanish the word equivalent to "HELLO" is "HOLA". In German, French, Italian and Russian language the word that means (or similar to) "HELLO" is "HALLO", "BONJOUR", "CIAO" and "ZDRAVSTVUJTE" respectively.

In this problem your task is pretty simple. You will be given one of the six words mentioned above or any other word and you will have to try and detect the language it is from.

Input

Input file contains around **2000** lines of inputs. Each line contains a string **S**. You can assume that all the letters of the string are uppercase English letters and the maximum length of the string is **14**. Input is terminated by a line containing a single **'#'** character (without the quote). This line should not be processed.

Output

For each line of input except the last one produce one line of output.
This line contains the serial of output followed by a language name. If the input string is "**HELLO**" or "**HOLA**" or "**HALLO**" or "**BONJOUR**" or "**CIAO**" or "**ZDRAVSTVUJTE**" then you should report the language it belongs to. If the input string is something other than these 6 strings print the string "**UNKNOWN**" (without the quotes) instead. All characters in the output strings are uppercase as well. Look at the output for sample input for formatting details.

Sample Input	Output for Sample Input
HELLO HOLA HALLO BONJOUR CIAO ZDRAVSTVUJTE #	Case 1: ENGLISH Case 2: SPANISH Case 3: GERMAN Case 4: FRENCH Case 5: ITALIAN Case 6: RUSSIAN

C Scientific Experiment

Input: Standard Input
Output: Standard Output

John wants to be a scientist. A first step of becoming a scientist is to perform experiment. John has decided to experiment with eggs. He wants to compare the hardness of eggs from different species. He has decided to use a nearby large multi-storied building for this purpose. For each species he will try to find the highest floor from which he can drop the egg and it will not break. The building has (n+1) floors numbered from 0 to n. John has a book from which he knows that

1. If an egg is dropped from floor 0, it will not break.
2. If an egg is dropped from the topmost floor, it will surely break.
3. The eggs of same species are of same strength. That means if any egg breaks when dropped from the **k**th floor; all the eggs of that species will break if dropped from **k**th floor.
4. If an egg is unbroken after dropping it from any floor, it remains unharmed, that means the strength of the egg remains same.

Unfortunately John has a few problems.

- He can only carry one egg at a time.
- He can buy eggs from a shop inside the building and an egg costs **x** cents.
- To enter the building he has to pay **y** cents if he has no egg with him and **z** cents if he carries an egg with him.
- After dropping an egg, John must go outside the building to check whether it's broken or not.
- He does not want to waste any egg so he will not leave any unbroken egg on the ground. But if an egg is broken, he leaves it there.
- If he has an intact egg at the end, he can sell it for **x/2** cents. He does not need to enter the building to sell the egg.

These problems are not going to tame John's curious mind. So he has decided to use an optimal strategy and minimize his cost in the worst case. As John is not a programmer, he has asked for your help.

Input

Input starts with a positive integer **T (T≤ 50)** denoting the number of cases.

Each case contains a line with 4 integer **n x y z** as described in the statement. You may assume that **1 < n ≤ 1000** and **1 ≤ x, y, z ≤ 10^5** and **x** is even.

Output

For each test case, print the case number and the minimized worst case cost.

Sample Input	Output for Sample Input
7 4 2 998 1000 16 2 1000 1000 16 1000 1 1 4 1000 1 1 7 2 2 2 9 2 1 100 11 2 100 1	Case 1: 2000 Case 2: 4008 Case 3: 1015 Case 4: 1003 Case 5: 10 Case 6: 24 Case 7: 111

Explanation of Case 1: John knows that the egg will break if dropped from 4th floor, but will not break if dropped from 0th floor. An optimal solution may be

- John enters the building without any egg (¢998).
- John buys an egg (¢2)
- John drops an egg from 2nd floor. John goes out and checks the egg.
    - If it breaks,
        - John again enters the building without any egg (¢998) and buys an egg there ¢2.
        - He drops the egg from 1st floor.
            - If it does not break then answer to his problem is 1 and he can sell the egg for ¢1. So his final cost in ¢1999.
            - If it breaks then the answer to his problem is 0th floor and his final cost is ¢2000.
    - If it does not break,
        - John enters the building with the egg (¢1000).
        - He drops it from 3rd floor.
            - If it does not break then answer to his problem is 3 and he can sell the egg for ¢1. So his final cost in ¢1999.
            - If it breaks then the answer to his problem is 2 and final cost is ¢2000.

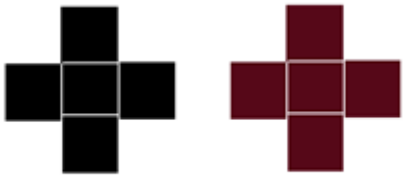
So, using this strategy, his worst case cost is ¢2000.

D ABC Tiles

Input: Standard Input
Output: Standard Output

Spring is coming! Tara is making a long list of the cleaning tasks that needs to be done as part of spring cleaning. One of the tasks is renovating the tiles of the kitchen wall.

The wall is a square of size **N x N** and was tiled with azure **N x N** tiles before few pieces fell off. Tara has ordered Soha to go out and get some azure tiles to cover those missing parts. Soha finds that the local shops have run out of azure tiles. However, they do have abundant supplies of black and crimson colored tiles. These tiles aren't the typical **1 x 1** square that we are accustomed to. Instead they are of the following shape:



BLACK CRIMSON

Soha bought **2000** tiles (**1000** of each type).
Given the configuration of the wall, can you help them decide whether it's possible to fill up all the empty spaces using the tiles Soha bought? For obvious reasons, a cell can't be occupied by more than one tile. To make the tiling more beautiful, they have added another constraint – no two cells that share an edge or a corner can be filled with the tiles of the same color. This rule doesn't apply to azure tiles, though. And of course, two cells that belong to the same tile will have the same color and so the above constraint doesn't apply here as well. The sample input/output should make things more clear.

Input

The first line of input is an integer **T(T≤1000)** that indicates the number of test cases. Each case starts with an integer **N (1≤N≤15)**. Each of the next **N** lines contains **N** characters each (giving you the configuration of the wall). Each character will either be **A** (meaning that cell is filled with an azure tile) or **.** (meaning that cell is empty).

There is a blank line before every case.

Look at the samples for more details on the format.

Output

For each case, output the case number first. If it's not possible to fill up all the empty spaces using the tiles meeting the constraints mentioned, print **`"Not Possible!"`** without quotes. If it is possible, output **N** more lines giving the new configuration of the wall. The empty spaces should be replaced by the first letter of the color of the tile that was used to fill that cell.
Since there could be multiple solutions, choose the one that comes lexicographically earliest. Lexicographical comparisons of two configurations should be done in row major order.

Suppose we have two string arrays **A** and **B**. **A** will come earlier than **B** if for some **x**, row($A[x]$) < row($B[x]$) and
row($A[i]$) == row($B[i]$) for i = 1 to x-1

Sample Input	Output for Sample Input
3 3 AAA AAA AAA 5 ...AA ...AA ...AA AAAAA AAAAA 7 AAAAA.A AA.A... A...A.A AA.AAAA A.AA.AA ......A A.AA.AA	Case 1: AAA AAA AAA Case 2: Not Possible! Case 3: AAAAABA AACABBB ACCCABA AACAAAA ABAACAA BBBCCCA ABAACAA

Illustration of the samples:
Case 1 – There are no empty cells. That is, it's already tiled.
Case 2 – It's not possible to fill up all the empty cells.
Case 3 – configuration shown on the right

E Simple Encryption

Input: Standard Input
Output: Standard Output

Some day in the future a person named Reuben wants to submit some interesting problems to the judging director of ACM ICPC (Association for Copotron Mechanics International Collegiate Programming Contest) World Finals so that they can be used in World Finals 2031. This contest requires problem submitters to submit their problems in encrypted format. But in year 2030, encryption software is not widely available on the Internet just to prevent terrorists from sending encrypted messages. So he wants to use a simplified encryption technique to submit his problem. He uses several 12-digit encryption keys to encrypt his messages. We would not disclose the encryption/decryption technique for safety because your next generations will be competing in World Finals by then. But Reuben also does not want to send his encryption key in a straightforward email. He wants to send a key (public) K_1 that will implicitly denote the actual key (private) $\mathbf{K_2}$. $\mathbf{K_2}$ is such a **12** digit number so that $\mathbf{K}_1^{K_2} \equiv \mathbf{K}_2$ (modulo $\mathbf{10^{12}}$). Given the value of $\mathbf{K_1}$ your job is to help Reuben find the value of $\mathbf{K_2}$.

Input

The input file contains around **1200** line of input. Each line contains an integer, which denotes the value of $\mathbf{K_1}$ **($0<K_1<50001$)**. A line containing a single zero terminates input.

Output

For each line of input produce one line of output. This line contains the serial of output, followed by the given public key $\mathbf{K_1}$ and then a possible actual private key $\mathbf{K_2}$. Look at the output for sample input for exact formatting. Inputs will be such that for given value of $\mathbf{K_1}$ there will always be at least one value of $\mathbf{K_2}$. Note that $\mathbf{K_2}$ should always have **12** digits and will not have any leading zeroes.

Sample Input	Output for Sample Input
78 99 0	Case 1: Public Key = 78 Private Key = 308646916096 Case 2: Public Key = 99 Private Key = 817245479899

F Electricity Connection

Input: Standard Inputr
Output: Standard Output

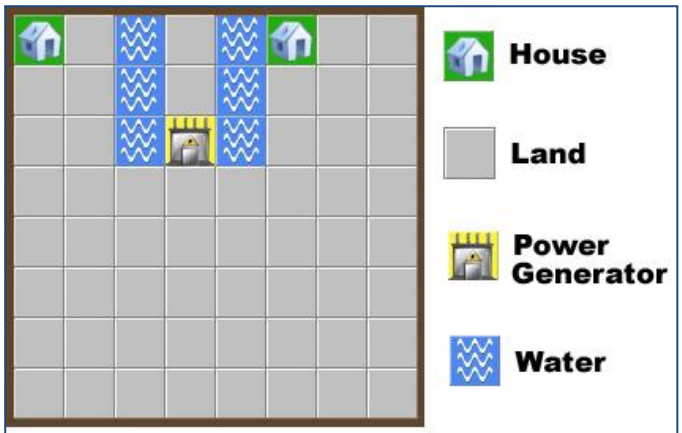
The city 'AjobakahD' has a lot of problems with electricity. Load shedding is a common problem here and people are quite used to it. Instead of calculating the total time the power is on, they calculate the total time the power is off. And of course the later one is always greater.

There is a small area in the city, which has not yet been enlightened with any load shedding! That means they haven't got the electricity connection yet. Now the Power Development Board (PDB) wants to set electricity connection in that area. Since the overall power in that city is not sufficient, they have decided to build a power generator in that area and want to connect all the houses to the generator.

The area can be modeled as an **8 x 8** grid. Each cell contains one of the following characters

'.' means land
'H' means house
'G' means power generator
'W' means water

Two adjacent cells can be connected by cables and the cost is **1** thousand. Two cells are said to be adjacent if they share a side. But two adjacent cells can only be connected if none of the cells is empty. Empty means either land or water. In such case, pillars can be built in the cells and after that they can be connected. The cost of placing a pillar in a land and water cell is **pl** and **pw** thousand respectively. Remember that both the costs can be zero, because there can be sponsors who might use the pillars to advertise themselves.



Example of an area

Now given the modeled grid of the area, the PDB wants to find the minimum cost to connect all the houses to the power generator directly or indirectly. That's why they seek your help, as you are one of the finest programmers in town.

Input

Input starts with a positive integer **T (T≤ 200)** denoting the number of cases.

Each case starts with two integers **pl** and **pw (0 ≤ pl, pw ≤ 10)**. Then there will be **8** lines, each containing **8** characters from the set **{., H, G, W}**. You may assume that in any modeled grid, there is exactly one power generator and the total number of houses is between **1** and **8** (inclusive).

Output

For each test case, print the case number and the total cost in thousands. Look at the output for sample input for formatting details.

Sample Input	Output for Sample Input
2 0 10 H.W.WH.. ..W.W... ..WGW... .... 0 0 H.W.WH.. ..W.W... ..WGW...	Case 1: 12 Case 2: 7

G Underwater Snipers

Input: Standard Input
Output: Standard Output

King *Motashota* is in a war against the mighty *Bachchaloks*. He has formed a well-trained army of snipers, and planning to use them as much as possible. In one of the missions, he has ***S*** snipers. They will be dispatched to get rid of the soldiers guarding the bank of the river '*Nodi*'.

From satellite images, *Motashota* has located positions of all enemy soldiers. Now, the plan is, snipers will take their positions. They are excellent swimmers, so, you can assume that they won't get caught, while taking position. Upon order from *Motashota*, they will start shooting enemy soldiers. A sniper can shoot a soldier, if euclidean distance between the soldier and sniper is no more than ***D***. After the snipers get rid of all the soldiers, they can proceed with the operation. So, it is important for them to position the snipers in such a way that, all soldiers are within the range of at least one sniper.

In addition, when snipers start shooting, the guards will be alert, and thus, snipers can't change their position, they can only continue shooting from their position.

The river bank is defined by the horizontal line $y = k$. All points (x,y) where $y>k$ is in the enemy territory, and if $y<k$, then it's on the water. You will be given location of ***N*** soldiers, strictly in the enemy territory, you have to place ***S*** soldiers in the water, so that, they can kill all soldiers. For security reasons, the snipers should be as far from the bank as possible. For any sniper in position (x_i,y_i), the distance from the bank is $|y_i-k|$. If, for all snipers, the minimum of them is $M=min\{|y_i-k|\}$, you have to maximize ***M***.

Both the soldiers and snipers are really superstitious. They will stay only in integer coordinates.

Input

First line contains an integer $T(1\le T\le 100)$, the number of test cases.

This is followed by ***T*** test cases. Each test case starts with four integers, $k(-10^8\le k\le 10^8)$, $N(1\le N\le 10000)$, $S(1\le S\le 10000)$ and $D(1\le D\le 10^9)$, the position of the bank, number of guards and number of snipers, and the range of the snipers.

This is followed by ***N*** lines, each containing a pair of integers (x_i, y_i) the position of i^{th} guard $(-10^8 \le x_i \le 10^8, k < y_i \le 10^8)$.

There is a blank line before each test case.

Output

For each test case, output the case number followed by an integer, ***M***, which is defined in the statement. If the snipers can't kill all guards, output: "IMPOSSIBLE".

Sample Input	Output for Sample Input
2 0 3 2 4 1 1 3 2 9 1 0 3 1 4 1 1 3 2 9 1	Case 1: 2 Case 2: IMPOSSIBLE

H Making Quadrilaterals

Input: Standard Input
Output: Standard Output

A quadrilateral is a simple geometric shape. The formal definition of Quadrilateral for this problem can be given as

"A quadrilateral is a simple polygon with four sides, having a strictly positive area."

If you are given four rods made of steel and having integer length, you may or may not be able to make a quadrilateral with it. For example you cannot make a quadrilateral with four rods of length 4, 5, 8 and 17 units but you can make a quadrilateral with four rods of length 2, 3, 4 and 5 units respectively. Now you have to supply n rods to the Architecture department of a University. But the University authority has asked you to make the length of the rods such that no four of them can be used to make a Quadrilateral. They are afraid that if the students can make such shapes then they will use up some of the rods in the sculptures they make. Given the value of n, what is the minimum possible length of the longest rod? You can assume that:

1. Only one rod has to be used as one side of the Quadrilateral.
2. A rod cannot be divided into two smaller pieces.
3. Two or more rods cannot be joined to make a longer rod.

Input

The input file contains around 100 line of input. Each line contains an integer, which denotes the value of n (3<n<61). A line containing a 0 (zero) terminates the input.

Output

For each line of input produce one line of output. This line contains serial of output followed by a decimal integer that denotes the shortest possible length of longest rod. You can safely assume that this length will fit in a 64-bit signed integer. Look at the output for sample input for details.

Sample Input	Output for Sample Input
4 6 0	Case 1: 3 Case 2: 9

Illustration of first Sample Input: If you have four sticks of length 1, 1, 1 and 3 then you cannot make a quadrilateral with them. So when n=4, the minimum possible length of the longest rod is 3.

I The Queue

Input: Standard Input
Output: Standard Output

On some special occasions Nadia's company provide very special lunch for all employees of the company. Before the food is served all of the employees must stand in a queue in front of the food counter. The company applied a rule for standing in the queue. The rule is nobody can stand anywhere in front of his supervisor in the queue. For example if Abul is the supervisor of Babul and Abul stands in k^{th} position from the front of the queue, then Babul cannot stand at any position in between 1 and k – 1 from front of the queue.

The company has **N** employees and each of them has exactly one supervisor except one who doesn't have any supervisor.

You have to calculate in how many ways the queue can be created. For this problem, you can safely assume that in at least one way the queue can be created.

Input

Input starts with an integer **T** (**T** is around **700**), the number of test cases.

Each test case starts with a line containing one integer **N (1 ≤ N ≤ 1000)**. Each of the following **N - 1** lines will contain two integers **a** and **b (1 ≤ a, b ≤ N and a ≠ b)**, which denotes that a is the supervisor of b. For the sake of simplicity we are representing each employee by an integer number.

Output

For each input case, output a single line in the format **"Case #: w"**, here **#** is the case number and w is the number of ways to create the queue. The number of ways can be very large. You have to print the number modulo **1,000,000,007**.

Sample Input	Output for Sample Input
1 5 2 1 2 3 3 4 3 5	Case 1: 8

J Air Pollution Problem

Input: Standard Input
Output: Standard Output

We the mankind have polluted the atmosphere all the time. That is why now in the year 2100 AD, people have to build cities covered with transparent glasses (Very hard glass though). This has to be done because people cannot survive in open air now and the artificial cover helps us to build an artificial atmosphere that is not polluted. But this system has not allowed us to remain 100% safe from the adverse atmosphere outside. Sometimes there are cracks in the glass cover and then people near that crack are supplied oxygen masks but not everyone in the city can be saved in that way. Whenever there is a crack in a certain part of the cover that part of the city is separated from the rest of the city with an invisible straight wall that prevents airflow from one part of the city to the other part, so the other part remains safe. But this creates another type of problem that will be explained somewhere below.

The city that we are talking about in this problem has 100 million people and they emit a lot of carbon-di-oxide (CO_2). The city is circular in shape (Can be considered an exact circle). As the citizens are continuously polluting the city atmosphere, many air purification plants are placed around the city in a scattered way. These plants add oxygen and remove carbon-di-oxide from the air. But the number of plants in the city is exactly according to the requirement of the city: Not much more than the required numbers. This is because (a) The establishment cost of plants is very high (b) Too many plants reduces the CO_2 level of air to a very low level, which can be harmful for the trees within the city (Yes there are very few trees in the city as well). So it may happen that when the city is divided into two parts one part may have less air purification plant than necessary and the other part may have more air purification plans than necessary. In other words inconsistency is created for the construction of the walls.

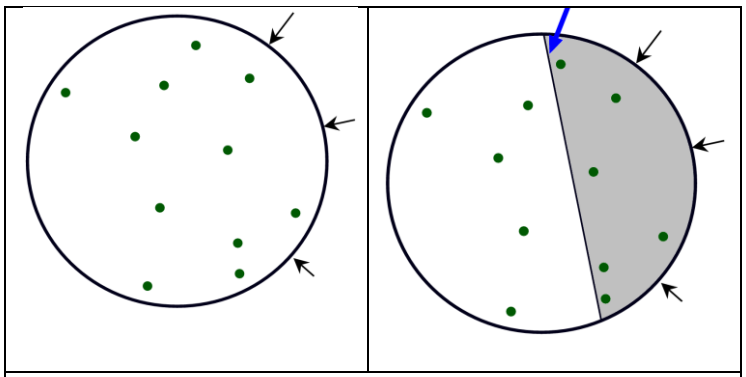


Figure: The city has 11 air-purification plant which are just enough for the requirement of the city. When the city is divided into two parts the left part has 5 and the right part has 6 air-purification plants. But the left part has larger area so it has less air-purification plants than required.

For example, in the figure on the left, the city (Denoted by a circle) has 11 air purification plants inside it (Denoted by small green dots). Cracks are seen in one side of the city wall (Marked with black arrow) and so a wall is instantly created (pointed with a blue arrow and denoted with e thin black straight line). Now suppose the area of the gray region is A_1 and area of the other region is A_2. The total city area is $A=A_1+A_2$. So the gray part ideally should have $\frac{A_1}{A}\times 11$ air purification plants but in reality it has 6 air purification plants. So inconsistency index for the gray part is $\left|\frac{A_1}{A}-\frac{6}{11}\right|$. Similarly, inconsistency

index for other part is $\left|\frac{A_2}{A}-\frac{5}{11}\right|$. The whole city can be divided into two parts in infinite possible ways. Given the coordinates of the air purifying plants in **2D** Cartesian coordinate system, your job is to find maximum possible inconsistency index considering all possible divisions. This will help the city designers, as they will try to place the air purifying plants in such locations so that maximum possible inconsistency is minimum. You can assume that the area that an air pollution plant occupies is zero, compared to the area of the total city and the dividing wall can be modeled as an straight-line (Zero thickness) .

Input

The input file contains maximum **220** sets of input. But most of the input cases are not extreme. The description of each set is given below:

First line of each set contains four integers **C_x, C_y, R (0<R ≤ 6000)** and **N (5 ≤ N ≤ 2000)**. Here **(C_x, C_y)** is the coordinate of the center of the city, **R** is the radius of the city and **N** is the total number of air purification plants in the city. Each of the next **N** lines contains two integers **(x_i, y_i) (0 ≤ x_i, y_i ≤ 10000)**, which denote location of one air purification plant. Any three purification plants will not fall on the same line.

Input is terminated by a line containing four zeroes. Most of the input cases are not extreme. There are only **6** cases where **N=500**, **4** cases where **N=1000** and only one case where **N=2000**. In all other cases **N ≤ 100**. The total size of input file is less than **150** kilo-byte and has less than **14000** lines in total.

Output

For each set of input produce one line of output. This line contains the serial of Scenario followed by a floating-point number. This floating-point number denotes the maximum possible inconsistency index and has six digits after the decimal point.

Sample Input	Output for Sample Input
4 5 8 8 5 6 0 5 8 4 2 2 4 9 0 8 4 4 3 9 0 0 0 0	Scenario 1: 0.339430