THE CODERS CUP

SENIOR ROUND

QUESTION SET

Competition Rules:

Participation Guidelines:

- The Freshmen Coders Cup round spans for 1 hour. If you have completed the problem set before the allocated time, you may leave the competition room quietly, but inform the coordinator must.
- If you leave the room, you cannot return.
- You can discuss only with your team mates. If you discuss with anyone else, both team will be disqualified from the competition.

Submission Guidelines:

- Find sample inputs from net-storage.
- The problem submission will be through PC^2
- Clarifications to any problem can be obtained using PC^2. No in room managers will be responsible for problems/confusions in problem set given.
- You are allowed to use language
    - C, C++, C#.NET, JAVA.
- IDEs allowed are:
    - Bloodshed Dev C++ for C and C++
    - Visual Studio 2008 or 2010 for C#.NET
    - NetBeans 6.8 for JAVA.
- Make console projects for all afore mentioned IDEs.
- Show output on console; don't write on a text file.
- Do not prompt for input from console in the program.
- Remove system ("pause")/getch()/package inclusion statements from your choice before submitting.

Additional Guidelines:

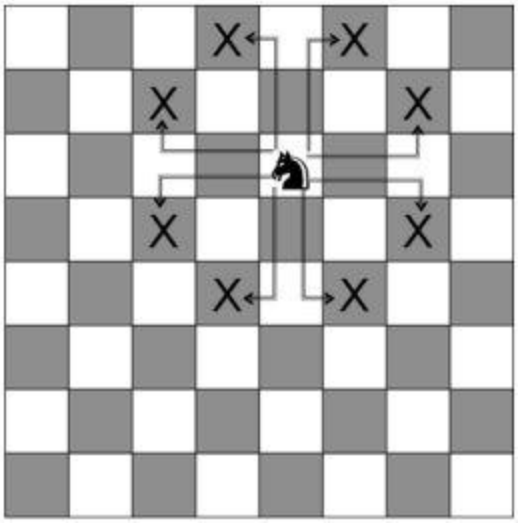
- The solution will be judged by multiple input files and execution time.
- The decision of judge will stand unchallenged.
- Books, manuals, and any sort of guide materials are not allowed.
- Your team can be dis-qualified, if found hard coding for solutions.
- Your team can be dis-qualified, if found using internet.
- Your team can be dis-qualified, if found unfair in anyway.

Note: Save your work continuously, ACM NUCES is not responsible for any loss of work due to power failure or any other reason.

Question 1:

A KNIGHT'S TALE

Chess is perhaps the oldest board game still played today. Each chess piece has its movement pattern. The most peculiar of these is the Knight. It travels either two steps horizontal and one step vertical, or two steps vertical and one horizontal, in each move.



This task requires you to find the smallest number of moves it takes for a knight to travel from one position to the next.

INPUT:

The input starts with a single integer T indicating the number of test cases. T test-cases follow after this line. Each test case contains two pairs of integers, one for the starting position, second for the destination. The first integer in each pair represents rows, and the second represents columns. The rows are numbered 0 to 7.

OUTPUT:

For each word in each test case, output the minimum number of moves it takes for the knight to get from start to end. Be sure to format it EXACTLY like shown in the sample output

Sample:

Input	Output
2 1 1 1 3 1 1 1 2	Case #1: 2 Case #2: 3

Question 2:

HAPPY NEW YEAR

Osama wants to send all of his contacts a personalized Happy New Year wish. Problem is, he has a lot of contacts. That means a lot of messages. Being the clever guy that he is, he comes up with a way to minimize his mails. He decides to use only the last or first name in each message, and thus group together people who have a name in common. Eg, John Doe, Jane Doe, Billy Buck, Billy Cyrus will only need 2 messages: One addressed to Doe and one addressed to Billy.

The task is to write a program that calculates the minimum amount of messages Osama has to send, given a list of contacts.

INPUT:

The input starts with a single integer T indicating the number of test cases. T test-cases follow after this line. Each test case starts with an integer N, defining the number of contacts in the list. The next N lines will each have the first and last name of the contact.

OUTPUT:

For each test case, output the minimum number of messages Osama has to send. Be sure to format it EXACTLY like shown in the sample output

Sample:

Input	Output
2 4 John Doe Jane Doe Billy Buck Billy Cyrus 3 Ali Ahmed Rafay Ahmed Shaffan Mehmood	Case #1: 2 Case #2: 2

Question 3:

GORO SORT

Goro has 4 arms. Goro is very strong. You don't mess with Goro. Goro needs to sort an array of N different integers. Algorithms are not Goro's strength; strength is Goro's strength. Goro's plan is to use the fingers on two of his hands to hold down several elements of the array and hit the table with his third and fourth fists as hard as possible. This will make the unsecured elements of the array fly up into the air, get shuffled randomly, and fall back down into the empty array locations.

Goro wants to sort the array as quickly as possible. How many hits will it take Goro to sort the given array, on average, if he acts intelligently when choosing which elements of the array to hold down before each hit of the table? Goro has an infinite number of fingers on the two hands he uses to hold down the array.

More precisely, before each hit, Goro may choose any subset of the elements of the array to freeze in place. He may choose differently depending on the outcomes of previous hits. Each hit permutes the unfrozen elements uniformly at random. Each permutation is equally likely.

EXPLANATION

In test case #3, one possible strategy is to hold down the two leftmost elements first. Elements 3 and 4 will be free to move. After a table hit, they will land in the correct order [3, 4] with probability 1/2 and in the wrong order [4, 3] with probability 1/2. Therefore, on average it will take 2 hits to arrange them in the correct order. After that, Goro can hold down elements 3 and 4 and hit the table until 1 and 2 land in the correct order, which will take another 2 hits, on average. The total is then 2 + 2 = 4 hits.

INPUT:

The first line of the input gives the number of test cases, T. T test cases follow. Each one will consist of two lines. The first line will give the number N. The second line will list the N elements of the array in their initial order.

OUTPUT:

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the expected number of hit-the-table operations when following the best hold-down strategy. Answers with an absolute or relative error of at most 10-6 will be considered correct.

Input	Output
3 2 2 1 3 1 3 2 4 2 1 4 3	Case #1: 2.000000 Case #2: 2.000000 Case #3: 4.000000

Question 4:

3-DIMENSIONAL MAZE

We have all written programs to solve mazes at some point in our CS College lives. Almost all of these mazes are 2-dimensional. This task asks you to try to extend this concept to 3-dimensions.

INPUT:

The input starts with a single integer T indicating the number of test cases. T test-cases follow after this line. Each test case starts with 3 integers X, Y, Z, defining the dimensions of the maze. After this, Z grids of X by Y characters will follow. A wall is represented by (*) and a path is represented (.) and the starting point is represented by a (S).

OUTPUT:

For each word in each test case, output the minimum number of steps needed to EXIT the maze. Diagonal moves are **NOT** allowed.

Input	Output
1 50 10 5 ********** ********** ********** ********** ********** ********** ********** ********** ********** ********** ********** *S.....****.....* ****.********** ****.********** ****.********** ****.....********** ********** ********** ********** ********** ********** ********** **********.....********** **********.********** **********.********** *********.....********** ********** ********** **********	Case #1: 40