Problem A
The World Responds
Input file: there is no input file for this problem

In many introductory computer programming classes, the first program that students learn to write just prints “Hello, world!” It is used as a first assignment because it is a simple program that produces output. The program dates back to at least 1974, when Brian Kernighan used it as an example in a C programming tutorial. Its popularity has grown since then. Louisiana Tech University's ACM chapter has a Hello World project that has collected versions of this program in almost 200 different computer languages.

While “Hello, World!” is a nice and simple first program, it does have one drawback. The program says hello to the world, but the world does not respond! This one-sided conversation is starting to become awkward after more than 35 years. For this problem, write a program that returns greetings from the world to the program.

Input

There is no input for this problem.

Output

Print one line of output: `The world says hello!`

Sample Input	Output for the Sample Input
	`The world says hello!`

Problem B

The Fastest Sorting Algorithm In The World

Input file: sorting.in

It is common to compare sorting algorithms based on their asymptotic speeds. Some slower algorithms like selection sort take $O(N^2)$ time to sort N items, while comparison-based sorts like merge sort can go no faster than $O(N \log(N))$ time, under reasonable assumptions. Bucket sort, which is not a comparison-based sort, can sort in $O(N)$ time. This is because bucket sort assumes that the range of possible values is small relative to N. In general, the speed of a sorting algorithm depends on the assumptions it can make about the data it is sorting.

One sorting algorithm that is often overlooked, despite its speed, is The Fastest Sorting Algorithm In The World. It sorts in O(1), or constant, time. Of course, the algorithm assumes that the input is an array that is already in fast-access memory and that the input is already sorted. For this problem, implement The Fastest Sorting Algorithm In The World.

Input

The input file contains multiple test cases, each of which describes an array to sort. Each array description starts with an integer $0 < N \leq 100$. N is followed by the N integers to sort, which will be given in non-decreasing order. All integers to sort are in the range of 0 to 100000. The last test case is followed by a line containing a single zero.

Output

For each test case, print the case number (beginning with 1) followed by the text `Sorting... done!`

Follow the format of the sample output.

Sample Input	Output for the Sample Input
5 21 44 48 48 64 6 8 19 22 49 53 62 8 5 9 14 17 24 25 27 61 4 13 21 28 35 5 31 38 44 49 60 0	Case 1: Sorting... done! Case 2: Sorting... done! Case 3: Sorting... done! Case 4: Sorting... done! Case 5: Sorting... done!

Problem C
Fikapaus
Input file: fika.in

In many cultures, the work day is punctuated by breaks. These breaks can improve productivity by allowing workers to relax. Some people like to drink water, some drink coffee, while others like to have a snack, read a book, take a walk, or talk to other people. In Sweden, the coffee break is popular and is called a "fikapaus."

Imagine that you work in Sweden and you want to take one fikapaus each day with your coworker. Your work requires both of you to travel independently around your city throughout the day. Because the travel plans change daily, the best time to meet for a fikapaus differs on different days. But you do know your travel schedules in advance, and you can plan on meeting when the two of you are closest together. Write a program that determines the best time for a fikapaus.

Input

Input consists of multiple test cases. Each test case describes the travel schedules for you and your coworker for one day. The travel schedules start with an integer $1 \leq D \leq 20$ indicating the number of destinations to which you and your coworker will each travel. Following this are D times and destination pairs, in the format of $M\ X_1\ Y_1\ X_2\ Y_2$. The integer $0 \leq M \leq 500$ represents the number of minutes after 9:00 AM when you will be at coordinates X_1, Y_1 in the city, and your coworker will be at coordinates X_2, Y_2 in the city. The coordinates are all integers in the range of 0 to 1000. For each test case the values for M are all unique and given in increasing order. The last test case is followed by a line containing a single zero.

Output

For each test case, print the case number (beginning with 1) followed by the time (in minutes after 9:00 AM) when you and your coworker will be closest to each other. Measure distance between points as the length of the straight line segment defined by the two points. If there are multiple times when the two of you will be closest, report the earliest time when you will be closest. Follow the format of the sample output.

Sample Input	Output for the Sample Input
4 99 62 44 66 20 199 14 36 67 92 261 4 35 62 22 359 15 31 11 55 6 0 21 32 31 6 87 23 62 77 76 134 59 25 56 81 180 36 50 29 25 288 69 9 83 16 411 81 61 65 3 0	Case 1: 99 Case 2: 288

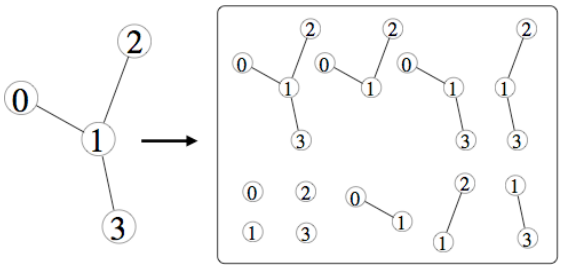
Problem D

How Many Subtrees?

Input file: subtrees.in

Trees are used for all sorts of purposes, such as parsing, information storage and retrieval, sorting, etc. An unweighted, undirected, unrooted tree T is made up of vertices and edges. Each edge connects two vertices. In a tree, any pair of vertices must be connected by some path of edges and vertices through the tree, but they may be connected by only one simple path (no cycles are permitted). Note that any tree with V vertices must have V-1 edges.

There are a lot of trees in computer science, but there are even more than might appear at first glance. This is because every tree is itself made up of one or more subtrees. A subtree S of a tree T is itself a tree, made only from the vertices and edges of T. A subtree must have at least one vertex, and a tree is considered a subtree of itself. Here is an example of a tree with four vertices (the large tree on the left) and its 11 subtrees (the smaller trees in the box on the right):



Write a program that, for each given tree, determines the number of subtrees that it has.

Input

The input file contains multiple test cases, each describing a tree. Each test case starts with an integer $1 \le V \le 10$ indicating the number of vertices in the tree. Vertices are implicitly labeled 0 through V-1. V is followed by V-1 edge descriptions. Each edge description has two integers $0 \le A < V$ and $0 \le B < V$, where $A \ne B$, indicating the endpoints A and B are connected. The last test case is followed by a line containing a single zero.

Output

For each test case, print the case number (beginning with 1) followed by the number of subtrees. Follow the format of the sample output.

Sample Input	Output for the Sample Input
4 0 1 1 2 2 3 4 0 1 1 2 1 3 0	Case 1: 10 Case 2: 11