Homework Assignment #3: Programming Exercise #1

Note

This assignment constitutes 4% of your grade and is due 2PM Thursday, May 14, 2009. Please write or type your answers/code on A4 (or similar size) paper. Drop your homework by the due time in Yih-Kuen Tsay's mail box on the first floor of Management College Building II. Late submission will be penalized by 20% for each working day overdue. You may discuss the problems with others, but copying answers/code is strictly forbidden.

Your work will be graded according to its correctness and presentation. Specifically, you should demonstrate evidences showing that your program is correct. You should also organize and document your program in such a way that your classmates, for example, can understand it.

Problem

Solve either Problem A "A Careful Approach" or Problem H "The Ministers' Major Mess" of the 2009 Annual ACM International Collegiate Programming Contest World Finals (see the appended).

Please prepare an input file with more interesting cases and test your program on the input. In the documentation of your program, you should describe how you have applied the algorithmic techniques, in particular design by induction, learned in class.

Problem A

A Careful Approach

Input: approach.in

If you think participating in a programming contest is stressful, imagine being an air traffic controller. With human lives at stake, an air traffic controller has to focus on tasks while working under constantly changing conditions as well as dealing with unforeseen events.

Consider the task of scheduling the airplanes that are landing at an airport. Incoming airplanes report their positions, directions, and speeds, and then the controller has to devise a landing schedule that brings all airplanes safely to the ground. Generally, the more time there is between successive landings, the "safer" a landing schedule is. This extra time gives pilots the opportunity to react to changing weather and other surprises.

Luckily, part of this scheduling task can be automated – this is where you come in. You will be given scenarios of airplane landings. Each airplane has a time window during which it can safely land. You must compute an order for landing all airplanes that respects these time windows. Furthermore, the airplane landings should be stretched out as much as possible so that the minimum time gap between successive landings is as large as possible. For example, if three airplanes land at 10:00am, 10:05am, and 10:15am, then the smallest gap is five minutes, which occurs between the first two airplanes. Not all gaps have to be the same, but the smallest gap should be as large as possible.

Input

The input file contains several test cases consisting of descriptions of landing scenarios. Each test case starts with a line containing a single integer n ($2 \le n \le 8$), which is the number of airplanes in the scenario. This is followed by n lines, each containing two integers a_i, b_i, which give the beginning and end of the closed interval $[a_i, b_i]$ during which the i^{th} plane can land safely. The numbers a_i and b_i are specified in minutes and satisfy $0 \le a_i \le b_i \le 1440$.

The input is terminated with a line containing the single integer zero.

Output

For each test case in the input, print its case number (starting with 1) followed by the minimum achievable time gap between successive landings. Print the time split into minutes and seconds, rounded to the closest second. Follow the format of the sample output.

Sample Input	**Output for the Sample Input**
3 0 10 5 15 10 15 2 0 10 10 20 0	Case 1: 7:30 Case 2: 20:00

Problem H

The Ministers' Major Mess

Input file: major.in

The ministers of the remote country of Stanistan are having severe problems with their decision making. It all started a few weeks ago when a new process for deciding which bills to pass was introduced. This process works as follows. During each voting session, there are several bills to be voted on. Each minister expresses an opinion by voting either "yes" or "no" for some of these bills. Because of limitations in the design of the technical solution used to evaluate the actual voting, each minister may vote on only at most four distinct bills (though this does not tend to be a problem, as most ministers only care about a handful of issues). Then, given these votes, the bills that are accepted are chosen in such a way that each minister gets more than half of his or her opinions satisfied.

As the astute reader has no doubt already realized, this process can lead to various problems. For instance, what if there are several possible choices satisfying all the ministers, or even worse, what if it is impossible to satisfy all the ministers? And even if the ministers' opinions lead to a unique choice, how is that choice found?

Your job is to write a program to help the ministers with some of these issues. Given the ministers' votes, the program must find out whether all the ministers can be satisfied, and if so, determine the decision on those bills for which, given the constraints, there is only one possible choice.

Input

Input consists of multiple test cases. Each test case starts with integers B ($1 \le B \le 100$), which is the number of distinct bills to vote on, and M ($1 \le M \le 500$), which is the number of ministers. The next M lines give the votes of the ministers. Each such line starts with an integer $1 \le k \le 4$, indicating the number of bills that the minister has voted on, followed by the k votes. Each vote is of the format <*bill*> <*vote*>, where <*bill*> is an integer between 1 and B identifying the bill that is voted on, and <*vote*> is either `y` or `n`, indicating that the minister's opinion is "yes" or "no." No minister votes on the same bill more than once. The last test case is followed by a line containing two zeros.

Output

For each test case, print the test case number (starting with 1) followed by the result of the process. If it is impossible to satisfy all ministers, the result should be `impossible`. Otherwise, the result should be a string of length B, where the i^{th} character is `y`, `n`, or `?`, depending on whether the decision on the i^{th} bill should be "yes," whether it should be "no," or whether the given votes do not determine the decision on this bill.

Sample Input	Output for the Sample Input
5 2 4 2 y 5 n 3 n 4 n 4 4 y 3 y 5 n 2 y 4 2 4 1 y 2 y 3 y 4 y 3 1 n 2 n 3 n 0 0	Case 1: ?y??n Case 2: impossible