Who has got the closest birthday?
(Solving the first day problem in code!)

The Problem

It's the first day of class and you are barely awake. You are hoping to snooze through a typical syllabus day when your new teacher commands you to get up and find the person in the room with the closest birthday to yours. Luckily, you've taken AP Computer Science already and realize that the key to solving the problem is to sort EVERYONE by their birthday, and then simply look directly to the left and right of you (the birthday that occurs immediately before and after yours) and see which of the two is closer. If you on the end of the list, you have to check with the person at the beginning and vice versa.

In order to solve this problem, you'll get several different classes from an input file. Each class will have several queries. You are required to implement either Merge Sort of Quick Sort in the solution of your assignment.

Input Specification

The input has a single positive integer, *n*, on its first line, specifying the number of classes in the input file.

The first line of input for each class will have a single positive integer *k (k < 1001)*, representing the number of students in the class. The next k lines will have information about each student. Each line will have the following information separated by spaces: first name, last name, month, day and year of birth. All names will only contain uppercase alphabetic characters and be no longer than 29 characters long. The month will be represented in its full spelling, in uppercase letters. The day and year will be the appropriate integers. You are guaranteed that all of this information is valid. (Thus, no April 31st will appear, etc.) It is also guaranteed that each full name will be unique. (Namely, no two people in a class will have the exact same first AND last name.)

Following those *k* lines will be a line containing a single positive integer *m (m < k)*, representing the number of queries for that class. The following *m* lines will contain the first and last name (separated by a space) of the student in question. (Your goal will be to find the name of the student with the closest birthday to the queried student.)

Output Specification

For each input class, print out a header with the following format:

```
Class #c:
```

where *c* represents the day of the simulation ($1 \leq c \leq n$).

Follow this with a blank line.

The following *k* lines will answer the queries for that class in the order they were given. For each of these queries, output a single line with the following format:

```
FIRST2 LAST2 has the closest birthday to FIRST LAST.
```

where FIRST LAST is the name of the queried student and FIRST2 LAST2 is the name of the student with the closest birthday to FIRST LAST.

To avoid ambiguity, sort the students in the following manner:

1) By birthdate, ignoring the year.
2) To break ties between students with the same exact birthdate, use last name as compared by the strcmp function.
3) To break ties where both #1 and #2 are the same, use the first name, which in these cases, is guaranteed to be different.

To further avoid ambiguity, if both the person who appears right before and right after the queried person are the same number of days away (in birthday) as the queried person, always choose the person who comes AFTER the queried person in the array. If the queried person is the LAST person in the array, then the person in index 0 will be considered as the person who comes AFTER them. Note: for these purposes, February 29th won't count as an actual day, unless someone in the class has that birthday. For example, if the queried person's birthday is March 1st, the person right before her has a February 28th birthday and the person right after her as a March 3rd birthday, then the person with the February 28th birthday is considered the closest (1 day way) as compared to the March 3rd birthday (2 days away). BUT, if there IS a February 29th birthday in the class, then that does count as a day.

Put a blank line between the output for each case.

Implementation Restrictions
You must fill in the given scaffold and implement Quick Sort.

All of the data for testing is posted on the course website.