Quiz Section for Program Design (II)

Exercise #8

In order to be a pokemon scientist, you decide to accept a big challenge from Professor Samuel Oak (大木博士). The pokemon Lab stores many data about Pokemon, but the data didn't sort well. It caused Professor Samuel Oak to spend a lot of time looking up the information. Professor Samuel Oak hopes you can help him design a system that can sort the data according to his requirements. He will hire you as a pokemon scientist if you can do it.

Professor Samuel Oak will give you **n** pokemon ($1 \leq n \leq 10^5$) and **m** queries ($1 \leq m \leq 10$). Each pokemon has its own name, attribute, attack value, and hp value. Then, there are four types of sorting requirements, "NAME", "ATTRIBUTE", "ATTACK", and "HP". The explanation of each sorting requirement is below.

Parameter	**Task**
NAME	Sort the pokemon according to their name. Use a dictionary order to sort the name from smallest to largest (we will guarantee that every pokemon's name is distinct in the test data). **Dictionary order denotes the way the words are ordered in a list, based on alphabetical order according to their alphabets. (ex: A < Z)**
ATTRIBUTE	Sort the pokemon according to the attribute. The priority of the attribute is WATER > FIRE > EARTH > LIGHT > DARK. The pokemon with a higher priority should be printed first than the pokemon with a lower one. If the two pokemon have the same attribute, which HP is less should be printed first.
ATTACK	Sort the pokemon according to the attack value from largest to smallest (we will guarantee every pokemon's attack value is different).
HP	Sort the pokemon according to the hp value from smallest to largest. (we will guarantee every pokemon's hp value is distinct)

Every time you finished the sorting according to the given requirement, you need to print "`Case #k: `" in the first line (**k** is from **1** to **m**). Then, you need to print the sorted pokemon in the format of "`NAME ATTRIBUTE ATTACK HP`". You **must** use the `qsort` from the `stdlib.h` in this exercise.

The table below shows the example input and output. The integers in the first line of input (e.g., `5 4` in the first example) represent the number of Pokemon (i.e., the value of **n**) and the number of sorting queries (i.e., the value of **m**). The next **n** lines of inputs represent every Pokemon's name, attribute, attack, and hp (assume that we have **n** pokemon). Then, the next **m** lines of inputs represent the sorting queries with different requirements.

Input	Output
5 4 FOODPAPA FIRE 1000 1000 LUNE WATER 500 10000 MEOWMEOW EARTH 50 50 KENN DARK 10000000 0 JIMMY LIGHT 150 33 NAME ATTRIBUTE ATTACK HP	Case #1: FOODPAPA FIRE 1000 1000 JIMMY LIGHT 150 33 KENN DARK 10000000 0 LUNE WATER 500 10000 MEOWMEOW EARTH 50 50 Case #2: LUNE WATER 500 10000 FOODPAPA FIRE 1000 1000 MEOWMEOW EARTH 50 50 JIMMY LIGHT 150 33 KENN DARK 10000000 0 Case #3: KENN DARK 10000000 0 FOODPAPA FIRE 1000 1000 LUNE WATER 500 10000 JIMMY LIGHT 150 33 MEOWMEOW EARTH 50 50 Case #4: KENN DARK 10000000 0 JIMMY LIGHT 150 33 MEOWMEOW EARTH 50 50 FOODPAPA FIRE 1000 1000

	LUNE WATER 500 10000
5 4 FOODPAPA FIRE 1000 1000 LUNE EARTH 500 10000 MEOWMEOW EARTH 50 50 KENN DARK 10000000 0 JIMMY FIRE 150 33 NAME ATTRIBUTE ATTACK HP	Case #1: FOODPAPA FIRE 1000 1000 JIMMY FIRE 150 33 KENN DARK 10000000 0 LUNE EARTH 500 10000 MEOWMEOW EARTH 50 50 Case #2: JIMMY FIRE 150 33 FOODPAPA FIRE 1000 1000 MEOWMEOW EARTH 50 50 LUNE EARTH 500 10000 KENN DARK 10000000 0 Case #3: KENN DARK 10000000 0 FOODPAPA FIRE 1000 1000 LUNE EARTH 500 10000 JIMMY FIRE 150 33 MEOWMEOW EARTH 50 50 Case #4: KENN DARK 10000000 0 JIMMY FIRE 150 33 MEOWMEOW EARTH 50 50 FOODPAPA FIRE 1000 1000 LUNE EARTH 500 10000