2013 Fall Computer Science I Program #1: Organ Transplant Database
Please consult Webcourse for the due date/time

The United States as an Organ Transplant database, to keep track of those who need organ transplants. Depending on a number of factors, when matches are found, organs are donated to those on the list. Typically, one can only receive a donation if the donor's organ is a match for the recipient. In this program, we will simulate a simplified organ transplant data base. To simplify the problem, our data base will simply keep track of the following information for each person in need of an organ:

1) Person's name
2) The organ they need to replace.
3) Their blood type (A+, A-, B+. B-, AB+, AB-, O+, O-)
4) The date (month, day, year) they were added to the list.
5) The time (hours, minutes) in military time they were added to the list.

Your program should use the following structs and constants:

```
#define SIZE 20
#define BLOODTYPESIZE 4

typedef struct {
    int month;
    int day;
    int year;
} dateT;

typedef struct {
    int hour;
    int minute;
} timeT;

typedef struct {
    char name[SIZE];
    char organname[SIZE];
    char bloodtype[BLOODTYPESIZE];
    dateT dateAdded;
    timeT timeAdded;
    int received;
} organT;
```

The name component will store the patient's name, organname will store the name of the organ they are in need of, bloodtype will store one of the 8 aforementioned strings representing bloodtype, the dateAdded and timeAdded will store when the corresponding organ was put on the waitlist while received will store a 0 initially and may store a 1 to indicate that the organ has been transplanted. (Note: In this assignment you will not physically remove the record of any patient, even if they have received their organ. This field will be solely used so that you can skip matching an organ to a patient who has already received one.)

Your program should read in information for a file consisting of the organ wait list, as well as a sequence of organs received for donation. For each organ received, your program should print out the name of the person and the organ they have received. The organ should go to the the person who has been on the waitlist the longest, who is a match for the organ. For the purposes of this assignment, a match occurs when the donor organ is the same AND the bloodtype of the donor is the same as the recipient. Once a match is found for an organ, they should not be matched again.

Input File Format

The first line of the input file will contain a single positive integer, *n (n ≤ 120000)*, representing the number of organs on the waiting list. The next *n* lines will contain information about one organ each. Each of these lines will contain the person's name, the organ they need replaced, their blood type, the date they were added to the organ database and the time they were added to the organ database. Each of these items will be separated by a space. All names will be comprised of letters and underscores only, all organ names will be comparised of lowercase letters, all bloodtypes will be one of the previously mentioned 8 strings, all dates will be of the format *m/d/y*, where *m, d,* and *y*, represent the numeric month day and year the patient was added to the organ donation list (for this particular organ). Finally, the time will be of the form *hr:min*, where *hr(0 ≤ hr ≤ 23)* and *min(0 ≤ min ≤ 59)* represent the numeric hour and minutes for the time the patient was added to the organ donation list. You are guaranteed that no two organs were added to the list on the same date and time and that no name or organ name will contain more than 19 characters.

The following line of the input file (line number *n+2)*, will contain a single positive integer, *k (k ≤ 1000)* representing the number of organs received, during some fixed period in time. The following *k* lines will contain information about the organs received, in the order they were received. Each of these lines will contain two strings separated by a space: the name of the organ and the blood type of the donor. These will both adhere to the specifications previously given.

Output Specification

Output a single line for each organ received. If a matching recipient exists in the database that hasn't yet received an organ, print out the name of the recipient, followed by the organ they received. If no match exists in the database, print out the following on a single line.

```
No match found
```

Sample Input and Output

This is posted separately in the files *organ.in* and *organ.out*.

Specification Details

You must dynamically allocate an array of the appropriate size of type organT to store all of the organs on the waiting list. You must free this memory once it is no longer in use. Though a fancier solution is possible, for this assignment you will be asked to simply run a linear search through the entire organ array to find the best match for each organ. Thus, your program will take a few seconds to run on a maximum sized input file.

Deliverables

Please turn in a single source file, *organ.c*, with your solution to this problem via Webcourses before the due date/time for the assignment. Make sure that your program reads form standard in and outputs to standard out, as previously shown in lab.