Problem 1 Compound-Data Sorting (4%)

In this problem you are to sort a number of personal data.

Input

The first line contains an integer n, $(1 \leq n \leq 100)$, the number of rows to be sorted.
Then n lines follow, each of which contains the data of the i^th person, which are his/her name s_i, (string with $1 \leq |s_i| \leq 100$), age a_i (integer), and weight w_i (integer).

Output

Print the data, one for each line, in the following order:

- The names are in ascending lexicographical order (字典順序).
- When two persons have the same name, the elder one should come first.
- When two have the same name and same age, the heavier guy comes first.

Example

Sample Input

```
5
Peter 16 40
Peter 30 60
Peter 30 100
Amy 24 45
Wong 80 80
```

Sample Output

```
Amy 24 45
Peter 30 100
Peter 30 60
Peter 16 40
Wong 80 80
```

Note

In this problem, it is easier to define a suitable *struct* data type for personal data. Then you can sort the data using built-in *qsort*$(\cdots)$ function.

To compare the alphabetical order of two strings, you can use the built-in *strcmp()* function.

Problem 2 Queries in Social Network (4%)

Use *struct* to accomplish the following task:

In this problem you are given the names of m persons, ($1 \leq m \leq 500$). The goal is to maintain a social network, so that the following two types of queries can be answered:

- `MakeFriend NameA NameB`

  This means that the person named `NameA` and the person named `NameB` become friends to each other after this query.

- `ListFriend NameA`

  This is a query for the friend list of `NameA` up to the present time. You need to output the list for this query.

Input
The first line contains one integer m, ($1 \leq m \leq 500$), the number of persons. Each of the next m lines contain one string s, ($1 \leq |s| \leq 20$), which is the name of each person.

The next line contains one integer n, ($1 \leq n \leq 1000$), the number of queries to be processed. Then there are n lines, each contains one query as described in the above format.

It is guaranteed that the name of each person consists of only alphabetical characters.

Output
For each query of type `ListFriend`, output the list friends of that person, separated by a space character ' ', in a line.

Example 1

Input

```
3
Amy
John
Xman
4
MakeFriend Amy John
MakeFriend Amy Xman
ListFriend Amy
ListFriend Xman
```

Output

```
John Xman
Amy
```

Problem 2 continued on next page. . .

Example 2

Input

```
5
Xman
Superman
Spiderman
Batman
Ironman
7
MakeFriend Xman Superman
MakeFriend Batman Superman
ListFriend Batman
ListFriend Ironman
MakeFriend Spiderman Superman
ListFriend Ironman
ListFriend Superman
```

Sample Output

```
Superman


Xman Batman Spiderman
```

Note

You may want to use the following *struct* for this problem:

```
struct person {
   char name[21];
   int list_of_friends[500];   // friend list

   int num_friends;     // current number of friends
   // or,
   int *end_of_list;     // pointer to the end of friend list
};
```