Social Network 1

Description

Users can create, join and quit groups in a social network. Sometimes, people are interested in knowing:

1. Which group has the maximum number of members.

We assume that people in the social network know each other if and only if they are in some common group. We say that two persons *A* and *B* know each other *via* Group *G*, if and only if both *A* and *B* are in Group *G*. People are also interested in knowing:

2. Who knows the maximum number of people in the social network.

In this Lab exercise, we will implement such a system that answers the above two types of questions, as well as simulates the users' creating, joining and quitting groups.

Input

The first line of the input contains an integer N (N<=500), which is the number of operations.

Each of the N lines followed is in one of the following forms:

1. *Createjoin A G.* Person *A* joins Group *G* (and joins it). If *G* does not exist yet, then *A* creates *G* first and then joins it. *A* and *G* are names (strings).
2. *Quit A G*. Person *A* quits Group *G* that he is currently in.
3. *Query 1*. The first type of queries. The system should output the name of the group which has the maximum number of members. If multiple names exist, the system should output the one which is lexicographically the smallest.
4. *Query 2*. The second type of queries. The system should output the name of a person who knows the maximum number of people in the social network. If multiple names exist, the system should output the one which is lexicographically the smallest.

We assume that: first, there is no group at the beginning; second, all group names as well as person names are distinct. All characters in the names are '0'-'9', 'A'-'Z', or 'a'-'z'. Each name contains at most 20 characters.

Output

For each query in the input, output a line containing the corresponding name, which is the answer to that query.

Sample Input

18

Createjoin Acer Zur

Createjoin Weixiang Zur

Createjoin Shuhe Zur

Createjoin Weixiang Yuan

Createjoin Acer Yuan

Quit Weixiang Zur

Query 1

Query 2

Createjoin Zing Apple

Createjoin Weixiang Apple

Createjoin Qing Dota

Createjoin Weixiang Dota

Createjoin Zing Dota

Query 1

Createjoin Qing Yuan

Createjoin Qing Apple

Createjoin Acer Dota

Query 2

Sample Output

Yuan

Acer

Dota

Acer

You are advised to implement three classes: Group, Person, and SocialNetwork for this task, with the *int main* method in the *SocialNetwork* class.

```
#ifndef person_h
#define person_h
#include <cstdlib>
#include <string>
#include <vector>
using namespace std;
class Person{
private:
   string name;
   vector <int> groups;
public:
   Person();
   Person(string name);
   string getName();
   void joinGroup(int grp);
   void quitGroup(int grp);
   int getNumOfGroups();
   …
};
#endif

#ifndef group_h
#define group_h
```

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <vector>
#include "Person.h"
using namespace std;
class Group{
private:
   int id;
   string name;
   vector <Person *> members;
public:
   Group();
   Group(int id, string name);
   int getId();
   string getName();
   void addMember(Person *p);
   void delMember(string name);
   int getNumOfMembers();
   …
};
#endif

#ifndef social_network_h
#define social_network_h
#include <cstdlib>
#include <cstring>
```

```
#include <vector>
#include "Person.h"
#include "Group.h"
using namespace std;
class SocialNetwork{
private:
    vector <Group *> groups;
    vector <Person *> persons;
public:
    SocialNetwork();
    Person * addPerson(string name);
    Group * addGroup (string name);
    string answerQuery1();
    string answerQuery2();
    …
};

#endif
```