Championship

The Academy of Young Europeans (AYE) is organizing football championships across Europe. They want to hold multiple championships across Europe, with each championship having its own participants. Organizing multiple championships at the same time is not a simple task for the AYE. To ease the process of organizing these championships, the AYE wants to build the ECP (Electronic Championship Processor) that can help them in keeping the records of the standings in each of the championships being held. They want the ECP to have the PIE characteristics (powerful, intuitive, and efficient).

As one of the lead programmers hired by the AYE to build the ECP, you want to make sure that the PIE characteristics are achieved. You began your work by studying the championship format given by the AYE:

1. There are multiple championships that are managed by the ECP.
2. All championships follow the same rules and regulations.
3. Each team can only participate in one championship.
4. Each championship contains at least two teams.

The rules and regulations of the championship are as follows:

1. The championships are held in a league format, i.e. the team with the most number of points win. There are no “knockouts” in all the championships. The team that is ranked the highest is known to “top the table”.
2. In each match, there are only two possible outcomes: one team wins (and the other loses) and a draw. A winning team receives 3 (three) points from the match while both teams receive 1 (one) point in a draw. Losing teams do not receive a single point.
3. At the end of the competition, the team with the most number of points wins.
4. Should there be a tie (i.e. same number of points), the team with the most number of wins should be ranked higher than those who have less number of wins. If there is still a tie, the team with the lexicographically-smaller name will be ranked higher. The AYE believes that this is the fairest rule they can think of.

After reading the specifications above, you assemble your team of programmers to build the ECP. As a start, you are going to create a sophisticated backend system that is able to answer queries about the championships. The queries are as follows:

1. Simulate a match between two different teams in the same championship in which one team wins.
2. Simulate a match between two different teams in the same championship in which the result of the match is a draw.
3. Print the name of the team who is “top of the table” in a given championship. “Top of the table” means that the team is the highest-ranked team in the competition.
4. Print the team name that has played the most number of matches across all championships.
5. Print the top three teams’ names in a particular championship.

You are given a tight deadline by the AYE to build the ECP. You do not want to disappoint the AYE and want to protect your reputation as one of the best programmers in the world. Therefore, you do not want to waste any time and start building the ECP.

Good luck!

Input

The first line consists of three integers, C (1 <= C <= 3), T (2 <= T <= 20), and Q (10 <= Q <= 100), separated by a single space, denoting the number of championships being held, the total number of teams, and the total number of queries respectively. C lines follow. Each of the C lines contains the name of a championship. After that, T lines follow, each containing information about a team's name and the championship that it joins. The name of the team and championship are separated by a single space. It is guaranteed that all team names and championship names are unique and consist of only lowercase letters ('a' – 'z') and are single words of at most 20 characters. The next Q lines contain queries that you need to answer. The queries follow the following specification:

Query Type Input Format: `<QUERY_TYPE> <APPROPRIATE_PARAMETERS>`

1. `win WINNING_TEAM_NAME LOSING_TEAM_NAME`
   A match was played in which the team WINNING_TEAM_NAME beat LOSING_TEAM_NAME. It is guaranteed that both teams are valid teams that have been registered into a championship. If the two teams are not in the same championship, print "invalid matching". Otherwise, print the name of the championship that the teams are in.

2. `draw TEAM_NAME_ONE TEAM_NAME_TWO`
   A match was played in which the result is a draw between TEAM_NAME_ONE and TEAM_NAME_TWO. It is guaranteed that both teams are valid teams that have been registered into a championship. If the two teams are not in the same championship, print "invalid matching". Otherwise, print the name of the championship that the teams are in.

3. `top CHAMPIONSHIP_NAME`
   Print the team name that is on "top of the table" in the championship CHAMPIONSHIP_NAME. It is guaranteed that the championship with the name CHAMPIONSHIP_NAME exists in the ECP.

4. `max`
   Print the team name that has played the most number of matches across all championships. If there are more than one team that has played the same (maximum) number of matches, print the one that is lexicographically-smallest.

5. `final CHAMPIONSHIP_NAME`
   This query is guaranteed to be the last query and it is guaranteed that this will be the last query in all test cases. Print the names of the top three teams in CHAMPIONSHIP_NAME in one line, separated by a single space, starting from the first-placed team to the third-placed team. The ranking of teams follow the requirements defined in the problem description above. If there are less than three teams in the competition, print all team names starting from the highest-ranked team to the lowest-ranked team.[1]

Output

Print the result of the query as described in the input format above. The last line of the output should contain a newline character. For the last query, there is no whitespace after the last team's name.

[1] Hint: Although you can make your Team class implements the comparable interface and implement the `compareTo(…)` method, you can actually solve this last query (and the top query) without sorting. Hence, no sorting is actually required for this problem.

Sample Input

```
1 5 13
premierleague
manutd premierleague
arsenal premierleague
liverpool premierleague
mancity premierleague
chelsea premierleague
top premierleague
win manutd liverpool
win manutd mancity
draw arsenal chelsea
win mancity chelsea
top premierleague
draw arsenal chelsea
max
win liverpool arsenal
draw mancity liverpool
max
top premierleague
final premierleague
```

Sample Output

```
arsenal
premierleague
premierleague
premierleague
premierleague
manutd
premierleague
chelsea
premierleague
premierleague
arsenal
manutd
manutd liverpool mancity
```

Explanation

We first separate the queries by “match-related queries” and “information-related queries” and the results respectively:

```
top premierleague (a)               arsenal

win manutd liverpool                premierleague
win manutd mancity                  premierleague
draw arsenal chelsea                premierleague
win mancity chelsea (b)             premierleague

top premierleague                   manutd

draw arsenal chelsea (c)            premierleague

max                                 chelsea

win liverpool arsenal               premierleague
draw mancity liverpool (d)          premierleague

max                                 arsenal
top premierleague                   manutd

final premierleague (e)             manutd liverpool mancity
```

We will see how the table changes after a group of matches, denoted by the states (a, b, c, d, and e) above. At the initial state (a), the standings of the championship “premierleague” are the same as the lexicographical ordering of the team names, all with 0 point.

At state (b), the standings are as follows:

Rank	Team	P	W	D	L	Points
1	manutd	2	2	0	0	6
2	mancity	2	1	0	1	3
3	arsenal	1	0	1	0	1
4	chelsea	2	0	1	1	1
5	liverpool	1	0	0	1	0

manutd won against liverpool and mancity (+6 pts), mancity beat chelsea (+3 pts), and arsenal drew with chelsea (both +1 pt). The query is top, hence the output is manutd.

At state (c), the standings are as follows:

Rank	Team	P	W	D	L	Points
1	manutd	2	2	0	0	6
2	mancity	2	1	0	1	3
3	arsenal	2	0	2	0	2
4	chelsea	3	0	2	1	2
5	liverpool	1	0	0	1	0

arsenal drew with chelsea (both +1 pts). The query is max, hence the output is chelsea since chelsea has played the most number of matches (3).

At states (d) and (e), the standings are as follows:

Rank	Team	P	W	D	L	Points
1	manutd	2	2	0	0	6
2	liverpool	3	1	1	1	4
3	mancity	3	1	1	1	4
4	arsenal	3	0	2	1	2
5	chelsea	3	0	2	1	2

liverpool won against arsenal (+3 pts) and mancity drew with liverpool (both +1 pt). The queries were max and top. Now, 4 teams have played 3 matches, but arsenal is the output since it is the smallest lexicographically. The top is still manutd.

The final query comes when the standings are illustrated in the last table above. manutd is the champion with 6 points, while liverpool is second with 4 points. mancity and liverpool both have 4 points and the same number of wins (1), but since liverpool is lexicographically smaller compared to mancity, liverpool is second, waiting for another year to come to get the chance to win the title. arsenal lurks in fourth place and chelsea is at the bottom of the table.

Skeleton

You are given the skeleton file **ECP.java**.

Note: You should see a file with the following contents during your sit-in lab when you open the file. If you see an empty file, please go to the correct directory (`ex1`) and see if the skeleton file is located inside.

```java
import java.util.*;

public class ECP {
    public static void main(String[] args) {
        //define your main method here
    }
}

class Championship {
    //define the appropriate attributes, constructor, and methods here
}

class Team {
    //define the appropriate attributes, constructor, and methods here
}
```

Notes:

1. You should develop your program in the subdirectory ex1 and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You only need to modify the skeleton file, that is **you do not need to create a new file for each class.** All code should be inside the file given to you in the ex1 directory.
3. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm. You are free to define your own classes besides the ones given in the skeleton file.
4. You must (and need to) use OOP for this sit-in lab.
5. You are free to define your own methods.
6. Please be reminded that the marking scheme is:

Input	: 10%
Output	: 10%
Correctness	: 50%
Programming Style	: 30%, which consists of:

   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (incremental programming, proper modifiers [public / private]: 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%