Scoreboard – Data Structures Exam

A **scoreboard** system keeps a set of **users**, **games** and **score** achieved by users playing the games. Each **user** has **username** and **password**. Each **game** has **game name** and **game password**. **Scoreboard** hold sets of score achieved in set of games. Your task is to model the scoreboard system and design a **data structure to hold the scoreboard**. Write a program that executes a sequence of commands, given in the input (a single command at a line):

- **`RegisterUser username password`** – registers a user into the system. Usernames are unique so duplicates are not allowed. As a result the command prints "**User registered**" in case of success or "**Duplicated user**" in case the username already exists.
- **`RegisterGame gameName password`** – registers a game into the system. Game names are unique so duplicates are not allowed. As a result the command prints "**Game registered**" in case of success or "**Duplicated game**" in case the game name already exists.
- **`AddScore username userPassword gameName gamePassword score`** – adds given score (integer number) to given game for given user. In case of success, adds the score and prints "**Score added**" as command result. In case the user or game does not exists or the passwords do not match, the command prints "**Cannot add score**". A user can have **multiple scores** achieved in the same game.
- **`ShowScoreboard game`** – shows the **top 10 highest score** for the specified game in the following format:

```
#1 peter 1560
#2 yavor 1560
#3 maria 1400
…
```

  - When multiple users have the same score, order them by username.
  - When less than 10 score exist in the specified game, show them all.
  - When more than 10 score exist in the specified game, show the first 10 top score.
  - Always show no more than 10 score. Even when more than 10 users have the same highest score, show the first 10 of them (this might look unfair, but that's the life).
  - When no score exist for the specified game, show "**No score**" as result.
  - When the game does not exist, show "**Game not found**" as result.
- **`ListGamesByPrefix namePrefix`** – shows the **first 10 games** in the alphabetical order that **match the specified prefix**. E.g. the prefix "**ch**" matches the games "**chess**" and "**chicken**" but does not match the game "**bosch**". The games should be printed on a single line **separated by comma + space**, in **alphabetical order**. If more than 10 games match the specified prefix, print the first 10 from all matches in the alphabetical order. If no games match the specified prefix, print "**No matches**" as command result.
- **`DeleteGame gameName password`** – deletes a game from the system. When a game is deleted, all of its score are also deleted. As a result the command prints "**Game deleted**" in case of success or "**Cannot delete game**" is case the game does not exists.
- **End** – this is the last line in the input. Indicates the end of the command sequence. Prints nothing.

Input

The input data should be read from the console. The input data consists of several commands, each on separate line, ending with the command "**End**". Empty lines in the input should be ignored.

The input data will consist of valid commands in the above described format. There is no need to check its validity.

Output

The output should be printed on the console. It should hold the output from each command from the input.

Constraints

- All usernames, passwords, game names and name prefixes:
  - Consist of **Latin letters and digits**.
  - Have length in the range [1…100].
- All **score** are integers in range [0…10 000 000].
- All string matching operations are **case-sensitive**.
- Allowed working time for your program: **1.00 seconds** (at the judge environment).
- Allowed memory: **32 MB**.

Examples

Input Example	Output Example
RegisterUser peter p123	User registered
RegisterUser maria m123	User registered
RegisterUser maria DuplicatedMaria	Duplicated user
RegisterGame AngryBirds a123	Game registered
RegisterGame chess c123	Game registered
RegisterGame Chess c123	Game registered
RegisterGame AngryBirds DuplicatedAngryBirds	Duplicated game
AddScore peter p123 AngryBirds a123 15000	Score added
AddScore peter p123 AngryBirds a123 160	Score added
AddScore peter p123 AngryBirds a123 15000	Score added
AddScore peter p123 AngryBirds a123 12700	Score added
AddScore peter p123 AngryBirds a123 8300	Score added
AddScore peter p123 AngryBirds a123 60	Score added
AddScore peter p123 AngryBirds a123 30	Score added
AddScore maria m123 AngryBirds a123 15000	Score added
AddScore maria m123 AngryBirds a123 8000	Score added
AddScore maria m123 AngryBirds a123 450	Score added
AddScore maria m123 AngryBirds a123 60	Score added
AddScore maria m123 AngryBirds a123 8000	Score added
AddScore maria invalidUserPass AngryBirds a123 1000	Cannot add score
AddScore maria m123 AngryBirds invalidGamePass 1000	Cannot add score
AddScore invalidUser m123 AngryBirds a123 1000	Cannot add score
AddScore maria m123 InvalidGame f123 1000	Cannot add score
ShowScoreboard AngryBirds	#1 maria 15000 #2 peter 15000 #3 peter 15000 #4 peter 12700 #5 peter 8300 #6 maria 8000 #7 maria 8000 #8 maria 450 #9 peter 160 #10 maria 60
AddScore peter p123 chess c123 200	Score added
AddScore maria m123 chess c123 600	Score added
AddScore maria m123 chess c123 200	Score added
ShowScoreboard chess	#1 maria 600 #2 maria 200 #3 peter 200
ShowScoreboard InvalidGame	Game not found
RegisterGame Chain c123	Game registered
RegisterGame Chaconne c123	Game registered
ListGamesByPrefix Ch	Chaconne, Chain, Chess
ListGamesByPrefix ch	chess

ListGamesByPrefix a ListGamesByPrefix Cha ListGamesByPrefix An DeleteGame InvalidGame pass123 DeleteGame AngryBirds invalidPass DeleteGame AngryBirds a123 ShowScoreboard AngryBirds ListGamesByPrefix An ShowScoreboard chess ShowScoreboard Chess RegisterGame AngryBirds a123 ShowScoreboard AngryBirds End	No matches Chaconne, Chain AngryBirds Cannot delete game Cannot delete game Game deleted Game not found No matches #1 maria 600 #2 maria 200 #3 peter 200 No score Game registered No score

Evaluation

- **Passed tests** give **50%** of the score for this problem.
- When **all tests pass** (with no exception), this gives the other **50%** of the score.

Submissions

Submissions are accepted for automatic evaluation at the SoftUni judge system:
https://judge.softuni.bg/Contests/113/Data-Structures-Exam-13-Sept-2015.