**Noah's Ark**

Help out the Modern-day Noah in his quest to save the animal species from the extinction. The only difference between Old-day Noah, and Modern-day Noah is that Modern-day Noah wants more than two of each species on the ark, but still wants them to be a even count.

Write a program that keeps count of how many animals of each kind are on the ark and whether they are an even or not.

Oh by the way he wants the list sorted in alphabetical order. Jeez... picky much?

**Input**

* Read from the standard input
* On the first line there is the number *N* - lines of input to follow;
* On the next *N* lines, there are the animals' species.

**Output**

* Print to the standard output
* On separate lines print how many of each animal have boarded the ark, and weather there is an even amount of them.
  + *See sample tests for examples*

**Sample tests**

**Input**

5

fish

crocodile

moose

moose

bald eagle

**Output**

bald eagle 1 No

crocodile 1 No

fish 1 No

moose 2 Yes

# Inventory Manager

## Objective

Your task is to implement a system that helps a store owner manage his inventory. New items can be added and filtered in the market. Here are the commands that the owner needs implemented:

* **add {ITEM\_NAME} {ITEM\_PRICE} {ITEM\_TYPE}** – adds a new item to the system;
  + **ITEM\_NAME** is a character sequence and has to be unique;
  + **ITEM\_PRICE** is a positive floating-point number;
  + **ITEM\_TYPE** is a character sequence and does not have to be unique;
  + If the item is added successfully, print: **Ok: Item {ITEM\_NAME} added successfully**;
  + If an item with the given name already exists, print: **Error: Item {ITEM\_NAME} already exists**;
* **filter by type {ITEM\_TYPE}** – lists the first 10 items (sorted) that have type equal to **ITEM\_TYPE**;
  + If the given **ITEM\_TYPE** does not exist, print: **Error: Type {ITEM\_TYPE} does not exist**;
* **filter by price from {MIN\_PRICE} to {MAX\_PRICE}** – lists the first 10 (sorted) items that have **ITEM\_PRICE** in the given range, inclusive;
* **filter by price from {MIN\_PRICE}** – lists the first 10 items (sorted) that have a greater **ITEM\_PRICE** than the given, inclusive;
* **filter by price to {MAX\_PRICE}** – lists the first 10 items (sorted) that have a smaller **ITEM\_PRICE** that the given, inclusive;
* **end** – marks the end of the commands. No commands will follow.

All items that are listed by the **filter** commands must be printed in the following format:

Ok: {LIST\_OF\_ITEMS}

**LIST\_OF\_ITEMS** contains the filtered items, separated by a space and a comma (", ") and each item is represented as **ITEM\_NAME(ITEM\_PRICE)**. The list should be 10 or less products. The price is formatted up two decimal places. They must also be sorted by the following criteria:

* First by **ITEM\_PRICE**, ascending
* Then by **ITEM\_NAME**, ascending
* Lastly by **ITEM\_TYPE**, ascending

If **LIST\_OF\_ITEMS** contains no items, then print just:

Ok:

## Input

The input data is given at the standard input. It consists of a sequence of commands, each at a separate line, ending by the command **end**. The commands will be valid (as described in the above list), in the specified format, within the constraints given below. There is no need to check the input data explicitly.

## Output

For each command from the input sequence print at the standard output its result on a single line.

## Constraints

* All **ITEM\_NAME** and **ITEM\_TYPE** will consist of letters and digits only. No spaces are allowed.
* The total number of lines in the input will be in the range [1 … 50 000]

## Sample Tests

### Input

add CowMilk 1.90 dairy

add BulgarianYogurt 1.90 dairy

add SmartWatch 1111.90 technology

add Candy 0.90 food

add Lemonade 11.90 drinks

add Sweatshirt 121.90 clothes

add Pants 49.90 clothes

add CowMilk 1.90 dairy

add Eggs 2.34 food

add Cheese 5.55 dairy

filter by type clothes

filter by price from 1.00 to 2.00

add FreshOrange 1.99 juice

add Aloe 2.7 juice

filter by price from 1200

add Socks 2.90 clothes

filter by type fruits

add DellXPS13 1700.1234 technology

filter by price from 1200

filter by price from 1.50

filter by price to 2.00

filter by type clothes

end

### Output

Ok: Item CowMilk added successfully

Ok: Item BulgarianYogurt added successfully

Ok: Item SmartWatch added successfully

Ok: Item Candy added successfully

Ok: Item Lemonade added successfully

Ok: Item Sweatshirt added successfully

Ok: Item Pants added successfully

Error: Item CowMilk already exists

Ok: Item Eggs added successfully

Ok: Item Cheese added successfully

Ok: Pants(49.90), Sweatshirt(121.90)

Ok: BulgarianYogurt(1.90), CowMilk(1.90)

Ok: Item FreshOrange added successfully

Ok: Item Aloe added successfully

Ok:

Ok: Item Socks added successfully

Error: Type fruits does not exist

Ok: Item DellXPS13 added successfully

Ok: DellXPS13(1700.12)

Ok: BulgarianYogurt(1.90), CowMilk(1.90), FreshOrange(1.99), Eggs(2.34), Aloe(2.70), Socks(2.90), Cheese(5.55), Lemonade(11.90), Pants(49.90), Sweatshirt(121.90)

Ok: Candy(0.90), BulgarianYogurt(1.90), CowMilk(1.90), FreshOrange(1.99)

Ok: Socks(2.90), Pants(49.90), Sweatshirt(121.90)