Clothing Company

A clothing company wants to handle all the pieces of cloths of all its shops with a new abstract data type (ADT) with the following operations:

- Create(): Builder to create the clothing company without pieces of cloth.
- addShop(shop): It adds a shop to the company with a given name in the input parameter. If it already exists, it ignores the operation.
- addModel (shop, model, n): It adds n cloth pieces of a new cloth model to a shop. If the shop already had this model, it increments the number of pieces. If the shop does not exist, it ignores the operation.
- lastModels(shop, n): It returns a list of the "n" last models added to a given shop in inverse order. First, it should be the last model added. Second should be the second to the last model, and so on. If there are not enough last models, it only returns the available ones in this inverse order. If the shop does not exist, it returns an empty list.
- shops(): It returns a list of the shops **shorted alphabetically**.
- pieces(shop, model): It returns the number of pieces of cloth of a given shop and model. It returns 0 if either the model or the shop does not exist.

It is mandatory to define a C++ class for implementing this ADT. This ADT can use any of the ADTs of the virtual campus. The operations should be fairly efficient. You need to define a main function that uses the class of the new ADT for handling the input and generating the output.

Indicate the complexity cost of each operation explaining it, as a comment prior to each operation in the programming code.

Input

Each case will be represented with several lines, ended with a line with the word "end". Each line will represent any of the following operations:

- addShop <shop>: It adds a shop to the company, without printing anything
- addModel <shop>, <model>, <n>: It adds n cloth pieces of a new cloth model to a shop, without printing anything.
- lastModels <shop>, <n>: It prints the last models of a shop separated with commas and no spaces, in a new line. It just prints a break line if the shop does not exist.
- shops: It prints the shops of a company separated with commas and no spaces, in a new line.
- pieces <shop> <model>: It prints the number of pieces of a model for a given shop in a new line.

An empty case with no operations (just an "end" line) will represent the end of cases.

Output

The output of each case will be just the printed messages of all the operations without adding new extra line breaks, as each operation prints its break line if printing anything.

Example of input

```
addShop Fuencarral
addShop Salamanca
addShop Atocha
addModel Atocha CasualJeans 7
addModel Atocha BlueTShirt 3
addModel Atocha RedSkirt 9
addModel Atocha CasualJeans 10
lastModels Atocha 3
shops
pieces Atocha CasualJeans
end
addShop Atocha
pieces Atocha CasualJeans
pieces Atocha RedCap
addModel Atocha BlueJeans 4
lastModels Atocha 5
pieces Atocha BlueJeans
pieces Salamanca BlueJeans
end
end
```

Example of output

```
CasualJeans,RedSkirt,BlueTShirt
Atocha,Fuencarral,Salamanca
17
0
0
BlueJeans
4
0
```