# Problem 4 – GUnit

GUnit is a unit testing framework – it provides the user with the opportunity to create unit tests which are very important to big project programming. It is still in beta though and you are given the job to format a database with all the classes, for each class - its methods, and of each of its methods – its unit tests.

You will be given several input lines which will contain info about a class’s method’s unit test. All valid lines will be in the following format:

{class name} | {method name} | {unit test name}

The elements are separated by a space, a ‘|’ (vertical line) and another space.The valid class names, method names and unit test names can **only** contain English alphabet letters and digits, and **must** always start with a **capital letter**. All parameters must be at least **2** symbols long.

Any input line that does not follow the, specified above, format, should be **ignored**.

Your task is to save every unit test to its corresponding method and every method to its corresponding class in the database. If a class with the given name already exists you should add the new method with its test to it. If the method also exists in the given existing class, you should just add the new unit test to the corresponding method. If even the test is not new, you should do nothing.

There is also a specific way in which the classes, methods, and unit tests should be sorted. The classes should be ordered first by the amount of unit tests it has – descending, then by the amount of methods it has – ascending, and then alphabetically. The methods should be ordered by the amount of unit tests they have - descending, and then alphabetically. The unit tests should be ordered by the length of their names – ascending and then by alphabetically.

### Input

* The input will come in the form of input lines.
* When you receive the command “It’s testing time!” the input should stop and the output should start.

### Output

* The output is simple. You must print all classes in the following format:
* “{class name}:
* “##{method1 name}:
* “####{test1 name}
* “####{test2 name}
* “##{method2 name}:
* “{class2 name}:
* …”
* For more info see the example below.

### Constraints

* Every class will always have at least one method and every method will always have at least 1 test.
* Any input that does not consist only of what was specified as a valid format, is to be treated as invalid.
* All invalid input **must** be ignored.
* Allowed time/memory: 250ms/16MB

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Rectangle | CalculateArea | Bla  Rectangle | CalculateArea | Bla2  Rectangle | CalculateArea | Bla3  Rectangle | CalculateArea | Bla4  Circles | CalculateArea | Bla  Circles | CalculateArea | Bla2  Circles | CalculateArea | Bla3  Circles | CalculateArea | Bla4  ASd | ASd | ASd  It's testing time! | Circles:  ##CalculateArea  ####Bla  ####Bla2  ####Bla3  ####Bla4  Rectangle:  ##CalculateArea  ####Bla  ####Bla2  ####Bla3  ####Bla4  ASd:  ##ASd  ####ASd |