# Problem 4 – Angular Parser

AngularJS is a very reliable JavaScript framework which makes development of web applications easier.

You will be given several input lines containing angular elements. You need to format those elements and store them. The elements you need take note of, will come in the following format:

**$app=‘My Application’**

**$controller=‘My Controller’&app=’My Application’**

**$model=‘My Model’&app=’My Application’**

**$view=‘My View’&app=’My Application’**

The elements are all strings and can consist of any ASCII character, except apostrophes, because they indicate the name of the element itself. **$app** is a module, **$controller** is a controller, **$model** is a model, and **$view** is a view. You need to store **all modules**, for every module **its controllers**, its **models**, and its **views**. Two elements are considered **different** if their name **AND** their **module** are different.

It is possible that the module holding an element is **nonexistent**, but Angular Parser is smart! In that case Angular Parser should **keep** the information, until a module with that name is **created**, and when that happens it should assign the given element to it. If the **stored** elements do not get their modules created till the end of the input sequence, then Angular Parser **deletes** them.

Modules should be ordered by the **amount of controllers** they have, in descending order, and if there are modules with equal amounts, they should be ordered by **amount** **of models**, in ascending order. Controllers, models and views should be ordered in **alphabetical** order. If all the criteria fail for a given instance, the **order** is of **entrance**.

### Input

* The input data is passed to the first JavaScript function found in your code as an **array of strings**.
* Every string will contain information about angular elements, in one of the formats specified above.

### Output

* Print at the console a JSON string that holds the modules, for each module, its controllers, its models, and its views. All data should be ordered in the format specified above.

### Constraints

* The input will always be in the specified formats. There is no need to check it explicitly.
* The strings with names can consist of **any** ASCII character, except apostrophes.
* The names of controllers, models and views will always be unique, **in the scope of their module**.
* Allowed time/memory: 200ms/16MB

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| $app='MyApp'  $controller='My Controller'&app='MyApp'  $model='My Model'&app='MyApp'  $view='My View'&app='MyApp' | {"MyApp":{"controllers":["My Controller"],"models":["My Model"],"views":["My View"]}} |

|  |  |
| --- | --- |
| **Input** | **Output** |
| $controller='PHPController'&app='Language Parser'  $controller='JavaController'&app='Language Parser'  $controller='C#Controller'&app='Language Parser'  $controller='C++Controller'&app='Language Parser'  $app='Garbage Collector'  $controller='GarbageController'&app='Garbage Collector'  $controller='SpamController'&app='Garbage Collector'  $app='Language Parser' | {"Language Parser":{"controllers":["C#Controller","C++Controller","JavaController","PHPController"],"models":[],"views":[]},"Garbage Collector":{"controllers":["GarbageController","SpamController"],"models":[],"views":[]}} |