## Breakfast Robot

*It’s finally the future! Robots take care of everything and man has been freed from the mundane tasks of living. There is still worked to be done though, since those robots need to be programmed first - we may have robot chefs, but we do not yet have robot software developers.*

Your task is to write the management software for a breakfast chef robot - it needs to **take orders**, keep track of available **ingredients** and output an **error** if something’s wrong. The cooking instructions have already been installed, so your module needs to **plug into** the system and only take care of **orders** and **ingredients**. And since this is the future and food is printed with nano-particle beams, all ingredients are microelements - **protein**, **carbohydrates**, **fat** and **flavours**. The library of recipes includes the following meals:

* Apple - made with **1 carb** and **2 flavour**
* Lemonade - made with **10 carb** and **20 flavour**
* Burger - made with **5 carb**, **7 fat** and **3 flavour**
* Eggs - made with **5 protein**, **1 fat** and **1 flavour**
* Turkey - made with **10 protein**, **10 carb**, **10 fat** and **10 flavour**

The robot receives instructions either to **restock** the supply, **cook** a meal or **report** statistics. The input consists of one of the following commands:

* restock <microelement> <quantity> - increases the stored quantity of the given microelement
* prepare <recipe> <quantity> - uses the available ingredients to prepare the given meal
* report - returns information about the stored microelements, in the order described below, including zero elements

The robot is equipped with a quantum field storage, so it can hold an **unlimited quantity** of ingredients, but there is no guarantee there will be enough available to prepare a recipe, in which case an **error message** should be returned. Their availability is checked in the **order** in which they **appear** in the recipe, so the error should reflect the first requirement that was not met.

Submit a **closure** that returns the management function. The management function takes one parameter.

### Input

Instructions are passed as a **string argument** to your management function. It will be called **several times** per session, so internal state must be **preserved** throughout the entire session.

### Output

The **return** value of each operation is one of the following strings:

* Success - when restocking or completing cooking without errors
* Error: not enough <ingredient> in stock - when the robot couldn’t muster enough microelements
* protein={qty} carbohydrate={qty} fat={qty} flavour={qty} - when a report is requested, in a single string

### Constraints

* Recipes and ingredients in commands will always have valid names.

### Examples

|  |
| --- |
| Execution |
| let manager = solution();  manager("restock flavour 50"); *// Success*  manager("prepare lemonade 4"); *// Error: not enough carbohydrate in stock* |

|  |  |
| --- | --- |
| Input | Output |
| restock carbohydrate 10  restock flavour 10  prepare apple 1  restock fat 10  prepare burger 1  report | Success  Success  Success  Success  Success  **protein=0** carbohydrate**=4** fat**=3** flavour**=5** |

|  |  |
| --- | --- |
| Input | Output |
| prepare turkey 1  restock protein 10  prepare turkey 1  restock carbohydrate 10  prepare turkey 1  restock fat 10  prepare turkey 1  restock flavour 10  prepare turkey 1  report | Error: not enough protein in stock  Success  Error: not enough carbohydrate in stock  Success  Error: not enough fat in stock  Success  Error: not enough flavour in stock  Success  Success  **protein=0** carbohydrate**=0** fat**=0** flavour**=0** |