## Uno wild card, this would be a hoot | Card template, Uno cards, CardsUno wild card, this would be a hoot | Card template, Uno cards, CardsUno wild card, this would be a hoot | Card template, Uno cards, Cards03. Wild Card Game

*A fun and wild card game!*

*Draw cards and split them by type.*

Write a function named **draw\_cards** that takes a **variable number** of **arguments** (tuples)and **keyword arguments** (key-value pairs).

The function **receives information** about **monster cards** and **spell cards** and **returns** a formatted string.

The **arguments** will be **passed** as follows:

* The first **group of arguments** will be an **unknown** **number** of **tuples**
* the **first** element in the **tuple** is the **name** of the **card (string)** (every card will have a **unique name**)
* the **second** element is the **card type (string)** (every card can either have a "**monster" type** or "**spell"** type)
* The **second group** will be an **unknown number** of **keyword arguments (key-value pairs)**
* the **key** is the **name** of the **card (string)** (every card will have a **unique name**)
* the **value** is the **card type (string)** (every card can either have a "**monster"** or "**spell" type**)

After receiving the information and calling the function:

* You should keep track of the different **card types** in separate collections
* **Sort** the **monster** cards in **descending order** by **their name**.
* **Sort** the **spell** cards in **ascending order** by their **name**.
* **Format** the result as follows (each card info on a different line):
* Separate **monster** and **spell** cards under appropriate **headings**:
* First are the **monster** cards (**only if there are cards** from this specific type), each on a **different line** under the heading

**"Monster cards:"**

* Then the **spell** cards (**only if there are cards** from this specific type), each on a **different line** under the heading

**"Spell cards:"**

**In the end, return** the output as described below.

***Note: Submit only the function in the judge system***

### Input

* There will be **no input from the console**, just parameters passed to your function.

### Output

* The **output** should look like this (each string should be on a new line):
* **Prefix monster** names with **" \*\*\*"** - (**exactly two** spaces and **three** asterisks "**\***").
* **Prefix spell** names with **" $$$"** - (**exactly two** spaces and **three** dollar signs "**$**").

**"Monster cards:"**

**" \*\*\*{monster\_name1}"**

**" \*\*\*{monster\_name2}"**

**...**

**" \*\*\*{monster\_nameN}"**

**"Spell cards:"**

**" $$${spell\_name1}"**

**" $$${spell\_name2}"**

**...**

**" $$${spell\_nameN}"**

* **Important notes:**
  + You may **exclusively** draw either "**monster**" or "**spell**" cards. If so, **return** **only** the **type** you have. **Do not include** the **heading** for the **missing** card type in your formatted string.
* There are **exactly two intervals** **before** the **monster's** and the **spell's** **names**.

### Constraints

* The **arguments** willbealways **before the keyword arguments**.
* Each **tuple** will always contain the **monster's or spell's name** with its **type**.
* There always will be **at least** **one** **tuple** and **at least** **one** **keyword argument** with a **valid** **name** and a **valid** **type**. (Valid card types are "**monster**" and "**spell**")

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| print(draw\_cards(("cyber dragon", "monster"), freeze="spell",)) | Monster cards:  \*\*\*cyber dragon  Spell cards:  $$$freeze |
| print(draw\_cards(("celtic guardian", "monster"), ("earthquake", "spell"), ("fireball", "spell"), raigeki="spell", destroy="spell",)) | Monster cards:  \*\*\*celtic guardian  Spell cards:  $$$destroy  $$$earthquake  $$$fireball  $$$raigeki |
| print(draw\_cards(("brave attack", "spell"), ("freeze", "spell"), lightning\_bolt="spell", fireball="spell",)) | Spell cards:  $$$brave attack  $$$fireball  $$$freeze  $$$lightning\_bolt |