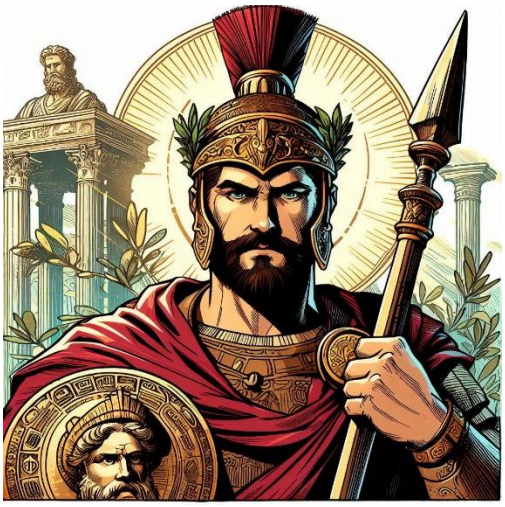
03. Roman Emperors



You are a student in History. As part of your final project on Ancient History, you are tasked to create a list of Roman Emperors, categorized by their status of rule as successful or unsuccessful.

Create a function named **`list_roman_emperors`** that **receives information** about various **Roman emperors**, their **success status**, and their **rule length** and **returns** a **sorted result** as **one formatted string**. It will receive an **unknown number** of **arguments** (**tuples**) and **keyword arguments** (**key-value pairs**). See the **examples** below.

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

- The **first group** of arguments will contain an **unknown number** of **tuples** with **two elements**
  - The **first element** represents the **Roman emperor's name** (a **valid string**)
  - The **second element** represents the **success status** of the **emperor** (a **Boolean** value)
- The **second group** will contain an **unknown number** of **keyword arguments** (**key-value pairs**)
  - The **key** will represent the **name** of the **emperor** (a **valid string**)
  - The **value** will represent the **length** of **their rule** (a **positive integer**)

After receiving the data and calling the function:

- You should **check each emperor's success status** (**True** or **False**) and **keep** the **information** about them in **separate collections** (you will have to **sort** them in the next steps)
- **Sort** the **successful** emperors **by** the **length** of their **rule** in **descending** order
  - If there are **emperors** with the **same length** of **rule**, **sort** them **alphabetically**, in **ascending** order
- **Sort** the **unsuccessful** emperors **by** the **length** of their **rule** in **ascending** order
  - If there are **emperors** with the **same length** of **rule**, **sort** them **alphabetically**, in **ascending** order
- The **first line** of your output string should contain the **total number** of **all emperors**:
  - **`"Total number of emperors: {num_of_all_emperors}"`**
- **Next**, arrange the **sorted data** under the **appropriate headings**:

- For **successful emperors** (**if any**), use the **heading** `"Successful emperors:"`
  - **Skip** the **heading** if there are **no successful emperors**
- For **unsuccessful emperors** (**if any**), use the **heading** `"Unsuccessful emperors:"`
  - **Skip** the **heading** if there are **no unsuccessful emperors**
- Note that you may receive **data either** for **successful or unsuccessful emperors** and your **output** should **contain sorted** information for the **received type only**. See the **examples** below.

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 **returned output** should look like this (**each** on a **new line**):

```
"Total number of emperors: {num_of_all_emperors}"
"Successful emperors:"
"****{emperor_name1}: {number_of_years}"
"****{emperor_name2}: {number_of_years}"
...
"****{emperor_nameN}: {number_of_years}"
"Unsuccessful emperors:"
"****{emperor_name1}: {number_of_years}"
"****{emperor_name2}: {number_of_years}"
...
"****{emperor_nameN}: {number_of_years}"
```

- **Prefix** the name of each emperor with **exactly four asterisks** `"****"`
- **Separate** the **name** of the emperor and his **years of rule** with a **colon** and a **single space** `": "`
- If you receive **only one type** of emperor (successful or unsuccessful), **return only** the type you've got. Do **not include** the **heading** for the **missing type** in your formatted string. See the **examples** below for clarification.

Constraints

- The **arguments** will always be **before** the **keyword arguments**.
- Each **tuple** will always provide a **unique name** (a **valid string**) of the emperor and his **status** (a valid **Boolean value**).

- Each **keyword argument** will always provide a **valid emperor's name** (a **string** that **corresponds** to the **same emperor's name** that will be **in** one of the **tuples**) and his **length of rule** (a **positive integer**).
- You will always receive **at least one tuple** with the **emperor's name** and **status** and **at least one** valid **keyword argument** containing **the same emperor's name** as a **key**.

Examples

Input	Output

`print(list_roman_emperors(("Augustus", True), ("Nero", False), Augustus=40, Nero=14,))`	`Total number of emperors: 2`
	`Successful emperors:`
	`****Augustus: 40`
	`Unsuccessful emperors:`
	`****Nero: 14`

`print(list_roman_emperors(("Augustus", True), ("Trajan", True), ("Nero", False), ("Caligula", False), ("Pertinax", False), ("Vespasian", True), Augustus=40, Trajan=19, Nero=14, Caligula=4, Pertinax=4, Vespasian=19,))`	`Total number of emperors: 6`
	`Successful emperors:`
	`****Augustus: 40`
	`****Trajan: 19`
	`****Vespasian: 19`
	`Unsuccessful emperors:`
	`****Caligula: 4`
	`****Pertinax: 4`
	`****Nero: 14`

`print(list_roman_emperors(("Augustus", True), ("Trajan", True), ("Claudius", True), Augustus=40, Trajan=19, Claudius=13,))`	`Total number of emperors: 3`
	`Successful emperors:`
	`****Augustus: 40`
	`****Trajan: 19`
	`****Claudius: 13`