# 03. Guest Accommodation



*Your system should match guests to available rooms, balancing comfort and efficiency. While some guests may not find a room, you must prioritize the seamless and well-organized process!*

Write a function named **"accommodate"** that **receives information** about **guests** waiting for accommodation and the available **rooms**.

The function will receive an **unknown number of arguments and keyword arguments**:

* The **arguments** will be passed as follows:
  + **Guest groups** as integers in the range **[1-10]** inclusive. (See the [**Examples**](#_Examples))
* The **keyword arguments** will be passed as follows:
  + **Room** **number as a key** - a unique string in the format **"room\_{number}"**.
    - The number contains **exactly 3** **digits**, in therange **[100-999]** inclusive.
  + **Room capacity as a value** - an integer in the range **[1-6]** inclusive.

**Example: room\_101=5**

The program runs until **all groups of guests** have **tried to complete** the **check-in** process at the reception area.

The receptionist tries to accommodate the **groups of guests** in their **initial order**, **starting** with the **first group**, and **searches for** the **most suitable** **room**.

* **Best fit rule**: A room is considered a **best fit** if it has a **capacity equal to the number of guests**. If there is **no such room**, choose the one with the **smallest capacity** that is **greater than the guests' number**. If multiple rooms have the **same capacity**, pick the one with the **smallest room number**.

**Hint**: Order the available rooms by capacity ascending, then by room number ascending.

* Once a group is **accommodated in a room**, the **room is no longer available**.
* If **no suitable room** is **found** (the number of guests does not fit in any available room), the group remains **unaccommodated**.
  + **Keep track** of the **number of guests** with **unsuccessful accommodations**.

Once **all guests** have **tried to** **complete** the check-in process, the hotel software **displays** the result.

* If there are **any successful accommodations**, **sort** them by the **room number ascending**.
* **Return** the output as described below. (See the [**Output**](#_Output) section)

***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

* If there are **accommodations**, **return** them **sorted** by **room number** and in the following format:

**"A total of {total\_number\_of\_accommodations} accommodations were completed!**

**<Room {room\_number1} accommodates {guests1} guests>**

**<Room {room\_number2} accommodates {guests2} guests>**

**…**

**<Room {room\_numberN} accommodates {guestsN} guests>"**

**Note:** If the room is "**room\_101**" the number of the room is **101**.

* Otherwise, **return** the **message**:

**"No accommodations were completed!"**

The **output string** should also contain information for the **guests without accommodation** and the **empty hotel rooms if there are any**:

* If there are **guests** **without** **accommodation**, **return** a **message**:

**"Guests with no accommodation: {total\_number\_of\_unaccommodated\_guests}"**

* If there are **empty rooms**, **return** a **message**:

**"Empty rooms: {total\_number\_of\_empty\_rooms}"**

## Constraints

* The **arguments** will always be **integers** in the **range [1-10]** inclusive.
* The **keyword** **arguments** will always have a **key** as a **string** in the format **"room\_{number}"** and a **value** as an **integer** in the **range [1-6]** inclusive.
* The **room numbers** will be **unique** andcontain **exactly 3 digits** in therange **[100-999]** inclusive.
* There will always be **at least** **one** **argument** and **at least one** **keyword** **argument**.

## Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| print(accommodate(5, 4, 2, room\_305=6, room\_410=5, room\_204=2)) | A total of 3 accommodations were completed!  <Room 204 accommodates 2 guests>  <Room 305 accommodates 4 guests>  <Room 410 accommodates 5 guests> |
| print(accommodate(10, 9, 8, room\_307=6, room\_802=5)) | No accommodations were completed!  Guests with no accommodation:27  Empty rooms: 2 |
| print(accommodate(1, 2, 4, 8, room\_102=3, room\_101=1, room\_103=2)) | A total of 2 accommodations were completed!  <Room 101 accommodates 1 guests>  <Room 103 accommodates 2 guests>  Guests with no accommodation: 12  Empty rooms: 1 |