Problem 4. Agency

Input / Constraints

Our task is to make a database for an agency which sells apartments.

A **basic** apartment will have:

- **id - string**
- **rooms – integer number**
- **baths - integer number**
- **square_meters – floating point number**
- **price – floating point number**

There are **only two types of apartments** the agency can manage: LivingApartments and OfficeApartments. Both types has the same parameters as a basic.

You will start receiving input data in format:

*** LivingApartment("{id}", {rooms}, {baths }, {square_meters }, {price})**

*-if there is no fifth parameter you should print **"__init__() missing 1 required positional argument: 'price'"***

In this case you do not add the apartment in DB

*** OfficeApartment("{id}", {rooms}, {baths }, {square_meters }, {price})**

*-if there is no fifth parameter you should print **"__init__() missing 1 required positional argument: 'price'"***

In this case you do not add the apartment in DB

Is it possible a request for basic apartment to reach the database:

***Apartment ("{id}", {rooms}, {baths }, {square_meters }, {price})**

You must print **"Can't instantiate abstract class Apartment with abstract methods __str__"** and you must not add it in the DB.

The core difference between OfficeApartments and LivingApartments is that the **LivingApartment's price is for buying and these apartment could not be hired and the OfficeApartmnets could only be hired not bought**.

When you receive the command '**start_selling'** you should **stop** adding apartments to the DB. From this moment you will **start receive a commands** in format:

{buy}/{rent} {id}

Where {id} is the id of the apartment that the client wants to rent/buy.

- **If the DB is not an apartment with the given id** print "Apartment with id - {id} does not exist!"
- If there **is such apartment**, but it is already **taken** print "Apartment with id - {id} is already taken!"
- If the apartment is available but the command is '**rent'** and the apartment **is of type LivingApartment** print "Apartment with id - {id} is only for selling!"
- If the apartment is available but the command is '**hire'** and the apartment **is of type OfficeApartment** print "Apartment with id - {id} is only for renting!"
- If **none** of the above cases are true, then just **mark the apartment as taken.**

Output

When you receive '**free**' or '**taken**' you should **stop selling** apartments and **make report**.

- If you have received the '**taken**' command you should print **just the taken apartments sorted first by price ascending and then by square meters descending**
- If you have received the '**free**' command you should print **just the free apartments sorted first by price descending and then by square meters ascending**

In both cases you should print the sorted apartments in the following format:

{rooms} rooms place with {bathrooms} bathroom/s.

{square_meters} sq. meters for {price} lv.

Where ***{rooms}*** are the count of the rooms in the apartment, ***{bathrooms}*** are the count of bathrooms in the apartments. ***{square_meters}*** – they are its square meters and ***{price}*** is the price .

- In case **there is no apartments** according the query print "**No information for this query**".

Examples

Input	Output	Comments

`Apartment("00A", 2, 1, 150, 100000)`
`OfficeApartment("00O", 2, 1, 500)`
`LivingApartment("00L", 3, 1, 180, 100000)`
`start_selling`
`rent 00L`
`taken`
	`Can't instantiate abstract class Apartment with abstract methods __str__`
	`__init__() missing 1 required positional argument: 'price'`
	`Apartment with id - 00L is only for selling!`
	`No information for this query`
		The first row is invalid because we get an Apartment and we can not add to the DB Apartment. It should be either Office/LivingAparment. We get valid kind on the second row, but the fifth param is missing, so we print the error on the console. And we do not add it to DB. On the third row everything is valid and we add to DB our fisrt and only aparment. We start selling and the only client we have want to rent an existing apartment, but the aparment is of LivingApartment kind so it is just for buying not for renting, so we print that on the console. We get 'taken' query, but we did not sell/ rent any aparments, so we just print that we do not have information about the 'taken' query.

Input	**Output**	**Comments**

`OfficeApartment("00O", 2, 1, 150, 500)`
`LivingApartment("00L", 3, 1, 180, 100000)`
`LivingApartment("01L", 3, 1, 190, 100000)`
`start_selling`
`buy 00L`
`buy 01L`
`rent 00O`
`taken`
	`2 rooms place with 1 bathroom/s.`
	`150.0 sq. meters for 500.0 lv.`
	`3 rooms place with 1 bathroom/s.`
	`190.0 sq. meters for 100000.0 lv.`
	`3 rooms place with 1 bathroom/s.`
	`180.0 sq. meters for 100000.0 lv.`	