Beds, Closets and Bedrooms - 3

Make a module apartment with the following folder structure. **Use imports wherever necessary.**

```
apartments
|--bed.py
|--closet.py
|--bedroom.py
|--apartment.py
|--kitchen.py
|--bathroom.py
|--flat.py
|--__init__.py
```

Use the same 3 classes in Beds, Closets and Bedrooms - 1 with the following specifications and some changes in the **add_bed() and add_closet() functions only**:

1. The Bed Object has the following attributes:

length: length of the bed in feet
breadth: breadth of the bed in feet
year_made: Year in which the bed was made
has_headboard: True or False depending on whether the bed has a headboard or not
has_posts: True or False depending on whether the bed has sideposts or not
material: material is wood, steel, plywood and so on.

2. The Bed Object does not support any following methods

___

1. The Closet Object has the following attributes:

length: length of the closet in feet
breadth: breadth of the closet in feet
height: breadth of the closet in feet
max_capacity: Total number of items that a closet supports
items: The list of items in the closet. [All strings]

2. The Closet Object supports the following methods:

store_item(): Takes a **string as input** and adds it to the items list
fetch_item(): Returns the frontmost object in the items list

____

1. The Bedroom object has the following attributes:

- **length**: length of the room in feet
- **breadth**: breadth of the room in feet
- **height**: breadth of the room in feet
- **bed:** an object representing the bed in the bedroom. **Initialize as None.**
- **closet:** an object representing the closet in the bedroom. **Initialize as None.**
- **has_balcony**: True or False depending on whether the room has a balcony or not
- **has_window:** True or False depending on whether the room has a window or not
- **num_lights:** The number of lights/lightsockets in the number

- **has_ac:** True or False depending on whether the room has a window or not
- **has_fan:** True or False depending on whether the room has a window or not
- **num_charging_points:** Number of charging points in the room.

2. The Bedroom object has the following methods:

- **carpet_area():** Returns the carpet area of the room which is calculated as length*breadth

- **add_bed():** creates a Bed object using user inputs [using **input()** function] and assigns it to the bed attribute of the bedroom. **While adding a bed make sure the dimensions of the bed are suitable for the remaining carpet area in the room.**

  For example: you cannot add a 9x9 bed in a 8X10 bedroom
  For example 2: you cannot add a 6x3 bed in a 8x10 bedroom if there is already a closet which takes up 60 sq ft space.

- **add_closet():** creates a Closet object using user inputs [using **input()** function] and assigns it to the closet attribute of the bedroom. **While adding a close make sure the dimensions of the closet are suitable for the remaining carpet area in the room.**

  For example: you cannot add a 9x9 closet in a 8X10 bedroom
  For example 2: you cannot add a 6x3 closet in a 8x10 bedroom if there is already a bed which takes up 60 sq ft space.

- **remove_bed():** Checks if the bed attribute is **None.** If not, then makes it None and returns “bed removed from the room”. If bed attribute is already None, then it returns “No bed found in the room”.

- **remove_closet():** Checks if the closet attribute is **None.** If not, then makes it None and returns “closet removed from the room”. If closet attribute is already None, then it returns “No closet found in the room”.

We have 3 new classes: Kitchen, Bathroom and Flats

1. The Kitchen has the following attributes:

length: length of the bed in feet
breadth: breadth of the bed in feet
slab_material: whether the slab is granite, wood, marble and so on.
has_sink: True or False depending on whether the kitchen has a sink or not
has_slab: True or False depending on whether the kitchen has a slab or not
furnishing_material: whether the material is wood, steel, plywood and so on.
lpg_pipeline: True or False depending on whether the kitchen has an LPG pipeline or not

2. The Kitchen Object supports the following methods:

cook(): Checks if lpg connection, slab and sink exist and returns “Kitchen can be used for cooking” . If these connections donot exist, returns “Kitchen unsuitable for cooking”

---

1. The Bathroom Object has the following attributes:

length: length of the closet in feet
breadth: breadth of the closet in feet
has_sink: True or False depending on whether the bathroom has a slab or not
has_bathtub: True or False depending on whether the bathroom has a bathtub or

not
has_tap: True or False depending on whether the bathroom has a tap or not
has_shower: True or False depending on whether the bathroom has a shower or not

2. The Bathroom Object supports the following methods:

bathing(): checks if atleast any one of the tap, shower or sink are available and returns “Suitable for bathing”, if not available it returns “Unsuitable for bathing”

---

1. The Flat has the following attributes:

bed_rooms: a list of all the bedrooms in the house, initialize as empty list
bath_rooms: a list of all the bathrooms in the house, initialize as empty list
kitchens: a list of all the kitchens in the house, initialize as empty list
owner_name: name of the flat owner, **initialize as None**
current_renter: name of the current renter, **initialize as None**
has_parking_permission: Initialize as False

2. The Flat Object supports the following methods:

rent_out(): Checks if flat is already on rent, if not then it returns the rent of the flat which is calculated as 5*carpet_area per month. Then it asks the user whether they agree to pay that amount or not using input(), if they say Y/Yes/yes then take another input() as their name and set the **current_renter** attribute. Return the rent value as well

PS: carpet area of the flat is the sum of carpet area of all the rooms in the house.

change_owner(): Takes a name as input from the user and changes the owner of the flat to that person

---

1. The Apartment has the following attributes:

flats: list of all flats
builder_name: name of the builder who built the apartment
amneties: list of all amneties in the apartments
parking_spots: number of parking spots available
number_floors: number of floors in the building
maintenance_monthly: monthly maintenance to be paid to the society
has_elevator: True or False depending on whether the building has an elevator
num_stairs: Number of flights of stairs in the building
fire_safety: True or False depending on whether the building has fire safety certification.

2. The Apartment Object supports the following methods:

rent_flat(): Takes the first unrented flat from the list of flats and calls its rent_out() function. To the returned value add 500 if the building has lift facility and an extra 500 if fire safety measures are present. Add to this 500 for each of the amneties in the apartment and also adds the monthly societal maintenance to return the final rent.

issue_parking_spot(): Issues a new parking spot if one is available

revoke_parking_spot(flat): takes a flat as input and revokes its parking permissions and makes the new parking spot available.