Project Outcomes:

Develop a python program that uses:

- decision constructs
- looping constructs
- basic operations on an list of objects (find, change, access all elements)
- more than one class and has multiple objects

Project Requirements:

1. Develop a simple **`Hotel`** program. We will have two classes, a **`Hotel`** class representing an individual hotel and a **`Room`** class. The **`Hotel`** class will contain several **`Room`** objects and will have several operations. We will also have a driver program to test the **`Hotel`** class.

2. Build a **`Hotel`** class that will store information about a Hotel. It will include a **name** and **location**. It should also include a **`list`** of class **`Room`** to hold information about each room. It will also have a int called **occupiedCnt** that keeps track of how many rooms in the hotel are occupied.

Specific Requirements for the `Hotel` Class:

1. The Hotel class has two constructors

    1. __init__ function, will read in the hotel name and location from hard-coded values in the tester class, such as Beach Marriot Pensacola, it will also assign numOfRooms to zero. numOfRooms indicates how many rooms are in the hotel. It will create a 10 element array.

2. The **`Hotel`** will have an **`addRoom`** method that will create each room with the required information: room number, bed type, smoking/non-smoking, and the room rate. Create at least 5 rooms with different characteristics. Each room will also have a boolean field called **`occupied`** attribute that will be set to false when the room is created. Don't forget to increment the numOfRooms instance variable. Example values for the rooms are:

```
101 queen s 100
102 king n 110
103 king n 88
```

```
104 twin s 100
105 queen n 99
```

3. The UML class diagram for the Hotel class will look like this:

Hotel
theRooms: Array Room[] name: String location: String occupiedCnt: int numOfRooms: int
def __init(self)__(String,String) def isFull(self) : boolean def isEmpty(self) : boolean def addRoom(self ,roomnumber,bedtype,smoking,price) def addReservation(self,occupantName ,smoking, bedtype) def cancelReservation(self,occupantName) def findReservation(self,occupantName): def printReservationList(self) def getDailySales(self) : def occupancyPercentage(self) : Setters and getters methods for name and location.

4. **`isFull()`** – returns a boolean that is true if all the rooms in the hotel are occupied.

5. **`isEmpty()`** – returns a boolean that is true if all the rooms in the hotel are unoccupied.

6. The **`addReservation()`** method takes three parameters: the occupant's name (String), smoking or non-smoking request (char), and the requested bed type (String). When this method is called, the hotel will search the list of its rooms for one that matches the bed type and smoking/non-smoking attributes. If an unoccupied room with the correct attributes is found, the renter's name will be set and the **`occupied`** attribute will be set to true. In either case a message will be printed that will state whether or not the reservation was made.

7. When the **cancelReservation()** method executes, the hotel will search for the name of the visitor in each room. If it is found, the occupied attribute will be set to false. In either case a message will state whether or not the reservation was cancelled. This method calls the private utility method **findReservation()** to scan the list of rooms looking for a guest by name. It will return the index of the room in the **Array** of rooms or **NOT_FOUND** if the room is not found, which will be declared as:

   ```
   NOT_FOUND = -1;
   ```

8. **findReservation()** will take in a String representing the occupant's name and search the occupied rooms for a reservation with that person's name. It will return the index of the room or **NOT_FOUND** if not found.

9. **printReservationList()** will scan through all the rooms and display all details for only those rooms that are occupied. For example:

   ```
   Room Number: 102
   Occupant name: Pinto
   Smoking room: n
   Bed Type: king
   Rate: 110.0

   Room Number: 103
   Occupant name: Wilson
   Smoking room: n
   Bed Type: king
   Rate: 88.0
   ```

10. **getDailySales()** will scan the room list, adding up the dollar amounts of the room rates of all occupied rooms only.

11. **occupancyPercentage()** will divide occupiedCnt by the total number of rooms to provide an occupancy percentage.

12. **__str__ –** returns a nicely formatted string giving hotel and room details (by calling the **__str__** in the **Room** class) for all the rooms in the hotel. For example:

```
Hotel Name : Beach Marriot
Number of Rooms : 5
Number of Occupied Rooms : 1


Room Details are:


Room Number: 101
```

```
Occupant name: Not Occupied
Smoking room: s
Bed Type: queen
Rate: 100.0

Room Number: 102
Occupant name: Coffey
Smoking room: n
Bed Type: king
Rate: 110.0

Room Number: 103
Occupant name: Wilson
Smoking room: n
Bed Type: king
Rate: 88.0

Room Number: 104
Occupant name: Not Occupied
Smoking room: s
Bed Type: twin
Rate: 100.0

Room Number: 105
Occupant name: Not Occupied
Smoking room: n
Bed Type: queen
Rate: 99.0
```

13. The **Room** class diagram will look like this:

Room
roomNum: int bedType: String rate: double occupantName: String smoking: char occupied: boolean

```
def __init__(int,String,char,double)
def getBedType(): String
def getSmoking(): char
def getRoomNum(): int
def getRoomRate(): double
def getOccupant(): String
def setOccupied(boolean)
def setOccupant(String)
def setRoomNum(int)
def setBedType(String)
def setRate(double)
def setSmoking(char)
def isOccupied(): boolean
```

1. The __init__() for a **Room** takes an int (room number), String (bed type), char (s or n for smoking or non-smoking)), and a double (room rate).

2. **isOccupied()** method returns true if the room is occupied, false otherwise.

3. **__str__()** provides all the details of a room - room number, name of guest(if occupied) , bed type, smoking/non-smoking, rental rate. This should all be formatted nicely with one attribute on each line using the '\n' escape character. See example above.

4. Several accessor and mutator methods for the **Room** class.

Use list to store the room details.

You have to store required data in the database. You can store hotel name, address, and all rooms. Customer data in database tables.