**Hotel Reservation**

**Explanation About Code:**

The HotelReservation class has methods to add and delete reservations, search tenants by ID, print tenant reports, print the list of confirmed tenants and waiting lists for a given reservation, and calculate the revenue. The main() function is the user interface that prompts the user to enter their choice of action to perform in the system.

The \_\_init\_\_() method initializes the reservations and tenants dictionaries as empty dictionaries.

The add\_reservation() method takes the parameters required to add a new reservation: tenant\_id, capacity, name, checkout\_date, checkin\_date, cost, and reservation\_number. It checks if the reservation\_number already exists and adds the reservation and tenant to the system.

The delete\_reservation() method takes a reservation\_number and deletes the corresponding reservation from the system.

The search\_tenant() method takes a tenant\_id and returns the tenant information for that ID.

The print\_report() method takes a tenant\_id and prints the tenant's information for that ID.

The print\_confirmed\_tenants() method takes a reservation\_number and prints the list of confirmed tenants for that reservation.

The print\_waiting\_list() method takes a reservation\_number and prints the waiting list for that reservation.

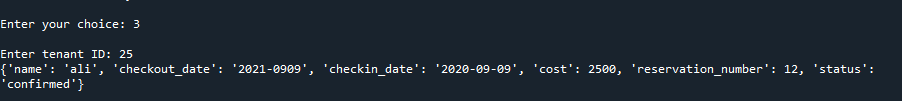
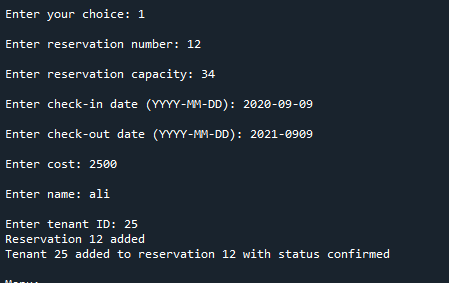
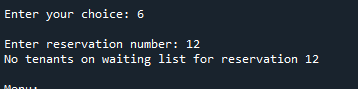
The calculate\_revenue() method calculates the total revenue for each reservation in the system.

The print\_total\_revenue() method prints the total revenue for each reservation in the system.

The main() function is the user interface that prompts the user to enter their choice of action to perform in the system. The user can add a new reservation, delete a reservation, search for a tenant using their ID number, print a report for a given tenant using their ID number, print the list of confirmed tenants for a given reservation, print the waiting list for a given reservation, print the total hotel revenue for each room reserved, or exit the system.

**Choice Of The Data Structure**

Using dictionaries as data structures was an appropriate choice for this system since they provide constant-time complexity O(1) for inserting, deleting, and searching elements based on their keys. Moreover, dictionaries allow us to access the values of their keys directly without the need for iteration or searching, making the system more efficient.

**Output**

