DESIGN SPECIFICATIONS

A) DATA STRUCTURES USED :

1) GUEST (Guest) :

Attributes :

- id (int) [Guest ID]
- priority (int) [Guest priority]
- stay_time (int) [Guest stay time in secs]

2) ROOM (Room) :

Attributes :

- room_id (int) [Room ID]
- guest (Guest) [Guest in the room]
- time_occupied (int) [Total time the room was occupied in secs]
- num_times_occupied (int) [Number of times the room was occupied]

3) HOTEL (Hotel) :

Attributes and semaphores:

- nondirty_and_empty_rooms (vector of Rooms) [vector of clean, empty rooms]
- dirtyRooms (vector of Rooms) [vector of dirty rooms]
- nondirty_and_occupied_rooms (vector of Rooms) [vector of clean, occupied rooms]
- net_occ_sem (sem_t) [semaphore to keep count of the net occupancy in hotel]
- clean_rooms_sem (sem_t) [SEMAPHORE to keep count of the no. of clean rooms]
- is_cleaning (bool) [flag to check if cleaning is in progress]

Methods :

1) Room getCleanRoom() :
    - Returns a clean, unoccupied room for a guest to stay in.

2) void occupy(Room &room, Guest guest) :
    - Does the necessary book-keeping when a guest is allotted a room.
    - If room occupancy = 1 on allocation, add room to vector of clean, occupied rooms.
    - If room occupancy > 1 on allocation, add room to vector of dirty rooms.

3) Room getLeastPriorityRoom() :
    - Returns the room with the least priority guest in it.

4) int cleanRoom(int& room_id) :
    - Cleans the room with the given room_id.
    - Returns the time taken to clean the room.

MUTEXES :

1) hotel_mutex (pthread_mutex_t) :
    - Used to lock the hotel object when a guest or cleaner thread is accessing it.

2) priority_lock (pthread_mutex_t) :
    - Used when a guest is being assigned a random priority (needed for assigning unique priorities to guests).

B) WORKING :

1) MAIN THREAD :

- The main thread creates a Hotel object.
- It initializes the clean_rooms_sem to n and the net_occ_sem to 2n.
- It also creates x cleaner threads and y guest threads.
- The guest threads indirectly run the guest(guest_id, priority) function defined in guest.cpp.
- The cleaner threads indirectly run the cleaner(cleaner_id) function defined in cleaner.cpp.

2) GUEST THREAD :

- The guest thread first creates a guest with given id and priority.
- The guest first sleeps for a random amount of time (10-20 s), and then tries to get a room to stay in for a random amount of time (10-30 s).
- If the hotel is currently being cleaned (checked using the value of `hotel.is_cleaning` flag), the guest won't get a room and has to try again after sleeping for some time.
- It then checks the number of clean rooms in the hotel using the value of `hotel.clean_rooms_sem` semaphore.
- If the semaphore value > 0, the guest gets a clean, unoccupied room using getCleanRoom() function, and can claim it straight away and occupy it for the stay time (The `clean_rooms_sem` is decremented).
- The `net_occ_sem` is also decremented for each such occupancy, and when it reaches 0, the hotel starts mass cleaning by evicting the guests and waking up the cleaner threads using `pthread_cond_broadcast`.
- Implementation-wise, the guest thread which first detects net occupancy = 0, sends SIGUSR1 signals to evict the other running guest threads which might have occupied some rooms, and sets the `hotel.is_cleaning` flag to true.
- Meanwhile, the suitable room, if obtained, is added to the clean, occupied rooms if `num_times_occupied`==1, else it gets added to the vector of dirty rooms.

- In the above scenario, the `clean_rooms_sem` is re-incremented only in the first case, hence there is a net decrement of 1 in the `clean_rooms_sem` value when the room finally gets added to the vector of dirty rooms.
- If a clean, empty room is unavailable, the guest looks for a room with the least priority guest in it using getLeastPriorityRoom() function, which looks for such a room in the set of clean, occupied rooms.
- If such a room is found, the guest evicts the guest in that room (again using SIGUSR1 signal) and occupies it for the stay time, and the necessary book-keeping is done as before.
- When the guest finishes with its stay, it leaves the room, allowing another waiting guest to occupy it.

3) CLEANER THREAD :

- The cleaner thread first checks if the hotel is currently being cleaned (by using `pthread_cond_wait` with the `hotel.is_cleaning` flag).
- When the cleaner thread is woken up by the one of the guest threads, it cleans a random room from the vector of dirty rooms.
- After cleaning, each such cleaner thread increments the `clean_rooms_sem` by 1 and the `net_occ_sem` by 2.
- When the `clean_rooms_sem` reaches n, the hotel is no longer being cleaned, and the `hotel.is_cleaning` flag is set to false.