Project 2 Design

A list of every semaphore, its purpose, and its initial value:

1. sem\_t guestMutex;
   * Purpose: This semaphore controls access to the guestQueue which manages the guests arriving at the hotel. It ensures that only one thread can access the guest queue at a time, preventing conflicts when adding or removing guests. This helps maintain the integrity of the guest queue.
   * Initial Value: sem\_init(&guestMutex, 0, 1);
2. sem\_t frontDeskMutex;
   * Purpose: This semaphore controls access to the front desk. It ensures that only one thread can access the front desk at a time, preventing conflicts when assigning rooms to guests. This helps maintain the integrity of the front desk operations.
   * Initial Value: sem\_init(&frontDeskMutex, 0, 1);
3. sem\_t bellhopMutex;
   * Purpose: This semaphore controls access to the bellhopQueue which manages the requests for bellhop assistance. It ensures that only one thread can access the bellhop queue at a time, preventing conflicts when adding or removing guests from the queue. This helps maintain the integrity of the bellhop queue.
   * Initial Value: sem\_init(&bellhopMutex, 0, 1);
4. sem\_t guestsReady;
   * Purpose: This semaphore indicates that guests are ready to be assigned a room. It allows the front desk employees to start registering guests once they are ready.
   * Initial Value: sem\_init(&guestsReady, 0, 0);
5. sem\_t bellhopRequested;
   * Purpose: This semaphore indicates that a bellhop has been requested by a guest. It allows the bellhop employees to start assisting guests with their bags.
   * Initial Value: sem\_init(&bellhopRequested, 0, 0);
6. sem\_t frontDeskEmployee;
   * Purpose: This semaphore lets two front desk employee work concurrently with check in of two guest.
   * Initial Value: sem\_init(&frontDeskEmployee, 0, 2);
7. sem\_t bellhopEmployee;
   * Purpose: This semaphore lets two bellhop employee work concurrently with the assisting of the guest with the bags.
   * Initial Value: sem\_init(&bellhopEmployee, 0, 2);
8. sem\_t guestLeaves;
   * Purpose: This semaphore indicates that a guest has left the front desk. It is used to synchronize the departure of guests towards their assigned room.
   * Initial Value: sem\_init(&guestLeaves, 0, 0);
9. sem\_t roomRegistration;
   * Purpose: This semaphore indicates that a room has been assigned to a guest by the front desk. It allows guests to proceed to their assigned rooms.
   * Initial Value: sem\_init(&roomRegistration, 0, 0);
10. sem\_t bagsDelivered;
    * Purpose: This semaphores is used to synchronize the delivery of bags between bellhops and guests.
    * Initial Value: sem\_init(&bagsDelivered, 0, 0);
11. sem\_t bagsReceived;
    * Purpose: This semaphores is used to synchronize the receiving end of bags between bellhops and guests.
    * Initial Value: sem\_init(&bagsReceived, 0, 0);
12. sem\_t\* checkedIntoRoom;
    * Purpose: This is an array of semaphores, one for each guest, to track whether each guest has reached their assigned room. It helps synchronize the process of guests entering their rooms.
    * Initial Value: checkedIntoRoom = new sem\_t[25];
13. sem\_t\* guestsRetired;
    * Purpose: This is an array of semaphores, one for each guest, to track whether each guest has retired for the evening. It helps synchronize the process of guests retiring for the night.
    * Initial Value: guestsRetired = new sem\_t[25];
14. sem\_t printSemaphore;
    * Purpose: This semaphore ensures that printing operations are thread-safe. It allows only one thread to print at a time to prevent overlapping output.
    * Initial Value: sem\_init(&printSemaphore, 0, 1);

2. Pseudocode for each function:

/\* Hotel Management System \*/

**semaphore** guestMutex = 1, frontDeskMutex = 1, bellhopMutex = 1;

**semaphore** guestsReady = 0, bellhopRequested = 0;

**semaphore** frontDeskEmployee = 2, bellhopEmployee = 2;

**semaphore** guestLeaves = 0, roomRegistration = 0;

**semaphore** bagsDelivered = 0, bagsReceived = 0;

**semaphore** printSemaphore = 1;

**semaphore** guestJoined = 0;

**int** guestCount = 0;

**int** current= 0;

**int** frontDeskEmployeeID[25] = {0};

**int** bellhopEmployeeID[25] = {0};

**semaphore** checkedIntoRoom [25] = {0};

**semaphore** guestsRetired[25] = {0};

**void** print(string message) {

semWait(printSemaphore);

message();

semSignal(printSemaphore);

}

**void** guest() {

guestCreated();

guestID = guestCount++;

randomBagAssign\_to\_guest();

enter\_hotel();

semWait(guestMutex);

guestID = count;

semSignal(guestMutex);

semWait(frontDeskEmployee);

semSignal(guestsReady);

semWait(guestsRetired[guestID]);

semWait(roomRegistration);

get\_room\_key(guestID);

semSignal(guestLeaves);

if (bag\_count > 2) {

semWait(bellhopEmployee);

request\_bellhop\_help(guestID);

semWait(bellhopMutex);

bellhopQueue.push(guestID);

semSignal(bellhopRequested);

semSignal(bellhopMutex);

semWait(bagsReceived);

enter\_room(guestID);

semSignal(checkedIntoRoom[guestID]);

semWait(bagsDelivered);

receive\_bags(guestID);

semSignal(bellhopEmployee);

} else {

semSignal(checkedIntoRoom[guestID]);

enter\_room(guestID);

}

retire(guestID);

guestJoin(guestID);

}

**void** frontDesk() {

frontDeskEmployeeCreated();

semWait(frontDeskMutex);

frontDeskID = get\_frontDesk\_ID();

semSignal(frontDeskMutex);

while (true) {

semWait(guestsReady);

semWait(frontDeskMutex);

Guest\* guest = guestQueue.front();

guestQueue.pop();

currentRoomNumber ++;

guest-> currentRoomNumber = currentRoomNumber;

semSignal(frontDeskMutex);

frontDeskEmployeeID[guest->id] = frontDeskID;

semSignal(roomRegistration);

guest\_is\_registered(guestID);

semSignal(guestsRetired[guest->id]);

semWait(guestLeaves);

semSignal(frontDeskEmployee);

}

}

**void** bellhop() {

bellhopEmployeeCreated();

semWait(bellhopMutex);

bellhopID = get\_bellhop\_ID();

semSignal(bellhopMutex);

while (true) {

semWait(bellhopRequested);

semWait(bellhopMutex);

Guest\* guest = bellhopQueue.front();

bellhopQueue.pop();

bellhopEmployeeID[guest->id] = bellhopID;

semSignal(bellhopMutex);

receive\_bags(guest->id);

semSignal(bagsReceived);

semWait(checkedIntoRoom);

deliver\_bags(guest->id);

semSignal(bagsDelivered);

semSignal(bellhopEmployee);

}

}