| **Semaphore** | **Initial Value** | **Purpose** |
| --- | --- | --- |
| frontDeskSlot | 2 | Capacity of the front desk. |
| bellhopSlot | 2 | Capacity of the bellhop. |
| guestReady | 0 | Signal for the waiting front desk employee. |
| finished | 0 | Signal for the customer that are waiting for the key room. Also use to signal for the customer that are waiting for bag delivery. |
| dropBag | 0 | Signal for the waiting bellhop. |
| takeBag | 0 | Signal for the waiting guess that the bellhop received the bags. |
| enterRoom | 0 | Signal for the waiting bellhop that the guess entered the room. |
| tip | 0 | Signal for the bellhop that waiting for a tip. |
| mutex1 | 1 | Protect access to the global queue guestQueue1. |
| mutex2 | 1 | Protect access to the global queue employeeQueue and roomQueue. |
| mutex3 | 1 | Protect access to the global queue guestQueue2. |
| mutex4 | 1 | Protect access to the global queue bellhopQueue. |

| semaphore frontDeskSlot = 2;  semaphore bellhopSlot = 2;  semaphore guestReady = 0, dropBag = 0, deliBag = 0, takeBag = 0, enterRoom = 0, tip = 0;  semaphore mutex1 = 1, mutex2 = 1, mutex3 = 1, mutex4 = 1;  semaphore finished[50] = {0}; | | |
| --- | --- | --- |
| void Guest() {  int id;  int bag = (int) (Math.random() \* 6);  int empId;  int keyNum;  int bellId;  enterHotel() ;  frontDeskSlot.acquire();  mutex1.acquire();  enqueue1(id);  guestReady.release();  mutex1.release();  finished[id].acquire();  mutex2.acquire();  dequeue2(empId);  dequeue3(keyNum);  receiveRoom();  mutex2.release();  if(bag > 2) {  requestHelp();  bellhopSlot.acquire();  mutex3.acquire();  enqueue4(id);  dropBag.release();  mutex3.release();  takeBag.acquire();  mutex4.acquire();  dequeue5(bellId);  entersRoom();  mutex4.release();  enterRoom.release();  finished[id].acquire();  receivesBags();  tip.release();  } else {  entersRoom();  }  retires();  } | void FrontDesk() {  int roomNum;  int id;  int fdGuest;  while(true) {  guestReady.acquire();  mutex1.acquire();  dequeue1(fdGuess);  mutex1.release();  mutex2.acquire();  enqueue2(id);  roomNum++;  enqueue3(roomNum);  registerGuest();  mutex2.release();  finished[fdGuest].release();  frontDeskSlot.release();  }  } | void Bellhop() {  int id;  int gid;  while(true) {  dropBag.acquire();  mutex3.acquire();  dequeue4(gid);  mutex3.release();  mutex4.acquire();  enqueue5(id);  mutex4.release();  getBag();  takeBag.release();  enterRoom.acquire();  deliBag();  finished[gid].release();  tip.acquire();  bellhopSlot.release();  }  } |