Akshat Sharma  
CSCI 474: Operating System Concepts  
Project 2: Semaphores and Threads  
Pseudocode

roomsAvailable[5];  
roomsOccupied[5] = {false};

Void guest(void \* guestNumber)

{

Wait(roomAvailable);  
 print “guest waits for check-in”;

Wait(availableCheckInReceptionist);

Print “guest goes to check-in receptionist”;  
 Signal(greetGuest);

Wait(assignRoom);  
Print “Guest receives room and completes check-in”;

Signal(availableCheckInReceptionist);

// Guest Activity –Sleep Code

//CheckOut code begins

Wait(availableCheckOutReceptionist);

Print “guest goes and return keys”;  
 Signal(greet\_checkOutGuest);

Wait(calculateBalance);  
Print “guest receives a total balance of $”;

Print “guest makes a payment of $”;  
Signal(guestMakesPayment);

Signal(availableCheckOutReceptionist);

}

Void checkIN()

{

Wait(greetGuest);  
Print “greet guest”;

Wait(roomAssignment);  
//Rooms Assignment Code  
//We need to perform Mutual Exclusion on the Shared Resources - RoomAssignment  
Signal(roomAssignment);

Print “Assign room to guest”  
Signal(assignRoom);

}

Void checkout()

{

Wait(greet\_checkoutGuest);

Print “greets guest and receives key”;

//Balance calculation code  
 Signal(calculateBalance);

Wait(guestMakesPayment);  
Print “receive $, complete check-out”;

Signal(roomAvailable);

}

Main()

{

//Initialize the semaphores here

guestThread[10];  
 checkInThread;  
 checkOutThread;

pthread\_create(checkInThread);  
 pthread\_create(checkOutThread);

for(I = 0; I < 10; i++)  
 {

pthread\_create(guestThreads);

}

for(I = 0; I < 10; i++)  
 {

pthread\_join(guestThreads);

}

pthread\_join(checkInThread);  
 pthread\_join(checkOutThread);

}