Semaphores:

|  |  |  |
| --- | --- | --- |
| Name | Purpose | Initial Value |
| patientMutex | This semaphore is to allow each patient thread to get a unique patient ID | 1 |
| nurseMutex | This semaphore is to allow each nurse thread to get a unique nurse ID | 1 |
| doctorMutex | This semaphore is to allow each doctor thread to get a unique doctor ID | 1 |
| queueMutex | This semaphore is to control the enqueue and dequeue of the nurseQueues | 1 |
| receptionistPatientMutex | This semaphore is to make sure a receptionist interacts with a single patient to get them registered | 1 |
| waitOnReceptionist | This semaphore is for the patient to wait for the receptionist to register the patient | 0 |
| startRegisteration | This semaphore is for the receptionist to wait for the patient to be ready to start the registration | 0 |
| patientSeated | This semaphore is for the receptionist to wait for the patient to be seated and then the receptionist can see the next patient | 0 |
| patientReady[3] | This semaphore is for the receptionist to inform the nurse that the patient it was assigned is ready to be taken to the doctor | 0 |
| doctorReady[3] | This semaphore is for the doctor to inform the nurse that it is ready to receive a new patient from the nurse | 0 |
| patientReceived[3] | This semaphore is for the doctor to wait to receive the patient from the nurse | 0 |
| doctorKnowsPatient[3] | This semaphore is for the nurse to wait to make sure the doctor knows the patient ID | 0 |
| patientInsideOffice[15] | This semaphore is for the doctor to wait for the patient to enter the office in order to begin the examination | 0 |
| appointmentDone[15] | This semaphore is for the patient to wait for the doctor to let the patient leave | 0 |
| patientEnterRoom[15] | This semaphore is for the patient to wait for the doctor to let it inside the room | 0 |
| giveAdvice[15] | This semaphore is for the patient to wait for the doctor to give advice | 0 |

Pseudocode:

semaphore patientMutex;

semaphore nurseMutex;

semaphore doctorMutex;

semaphore queueMutex;

semaphore receptionistPatientMutex;

semaphore waitOnReceptionist;

semaphore startRegisteration;

semaphore patientSeated;

semaphore patientReady[3];

semaphore doctorReady[3];

semaphore patientReceived[3];

semaphore doctorKnowsPatient[3];

semaphore patientInsideOffice[15];

semaphore appointmentDone[15];

semaphore patientEnterRoom[15];

semaphore giveAdvice[15];

int patientCount = 0;

int nurseCount = 0;

int doctorCount = 0;

int patientNumND[3];

int receptionistPatientNum;

int receptionistDoctorNum;

int nurseExit[3];

int doctorExit[3];

void patient(void \*counter)

{

counters \*c = (counters \*) counter;

int patientNum;

int doctorNum;

doctorNum = rand() % c->doctors;

wait(patientMutex);

patientNum = patientCount;

patientCount++;

signal(patientMutex);

printf("Patient %d enters waiting room, waits for receptionist\n", patientNum);

wait(receptionistPatientMutex);

receptionistPatientNum = patientNum;

receptionistDoctorNum = doctorNum;

signal(startRegisteration);

wait(waitOnReceptionist);

printf("Patient %d leaves receptionist and sits in waiting room\n", patientNum);

signal(patientSeated);

signal(receptionistPatientMutex);

wait(patientEnterRoom[patientNum]);

printf("Patient %d enters doctor %d's office\n", patientNum, doctorNum);

signal(patientInsideOffice[patientNum]);

wait(giveAdvice[patientNum]);

printf("Patient %d receives advice from doctor %d\n", patientNum, doctorNum);

wait(appointmentDone[patientNum]);

printf("Patient %d leaves\n", patientNum);

}

void receptionist(void \*counter)

{

counters \*c = (counters \*) counter;

int i = 0;

int nurseNum;

while(i < c->patients)

{

i++;

wait(startRegisteration);

printf("Receptionist registers patient %d\n", receptionistPatientNum);

nurseNum = receptionistDoctorNum;

wait(queueMutex);

enqueue(nurseQueue[nurseNum], receptionistPatientNum);

signal(queueMutex);

signal(waitOnReceptionist);

wait(patientSeated);

signal(patientReady[nurseNum]);

}

for(i = 0; i < c->nurses; i++)

{

nurseExit[i] = 1;

signal(patientReady[i]);

}

}

void nurse()

{

int nurseNum;

wait(nurseMutex);

nurseNum = nurseCount;

nurseCount++;

signal(nurseMutex);

while(!nurseExit[nurseNum] || !isEmpty(nurseQueue[nurseNum]))

{

wait(doctorReady[nurseNum]);

wait(patientReady[nurseNum]);

if(nurseExit[nurseNum] == 1 && isEmpty(nurseQueue[nurseNum]))

{

break;

}

wait(queueMutex);

patientNumND[nurseNum] = dequeue(nurseQueue[nurseNum]);

signal(queueMutex);

printf("Nurse %d takes patient %d to doctor's office\n", nurseNum, patientNumND[nurseNum]);

signal(patientReceived[nurseNum]);

wait(doctorKnowsPatient[nurseNum]);

}

doctorExit[nurseNum] = 1;

signal(patientReceived[nurseNum]);

}

void doctor()

{

int doctorNum;

int patientNum;

wait(doctorMutex);

doctorNum = doctorCount;

doctorCount++;

signal(doctorMutex);

while(!doctorExit[doctorNum])

{

signal(doctorReady[doctorNum]);

wait(patientReceived[doctorNum]);

if(doctorExit[doctorNum] == 1)

{

break;

}

patientNum = patientNumND[doctorNum];

signal(doctorKnowsPatient[doctorNum]);

signal(patientEnterRoom[patientNumND[doctorNum]]);

wait(patientInsideOffice[patientNum]);

printf("Doctor %d listens to symptoms from patient %d\n", doctorNum, patientNum);

signal(giveAdvice[patientNumND[doctorNum]]);

signal(appointmentDone[patientNumND[doctorNum]]);

}

}