Semaphores :

Receptionist_register ~

- Determines if the receptionist is available or occupied registering a patient
- Initially set to 1

Nurse_ready ~

- This semaphore indicates the amount of nurses that are available to walk the patients to their respective doctor offices
- Initial value = number of doctors

Doctor_ready ~

- Determines the availability of doctor threads being available to service patients
- Initial value = number of doctors

Patient_ready_receptionist ~

- This semaphore is used to signal from the patient to the receptionist that the patient thread obj is not available in the que line for the receptionist to access it.
- Initialized to 0

patient_ready_Nurse ~

- This semaphore signals to the nurse thread that there is a patient thread available in the queue line
- Initialized to 0

patient_ready_Doctor ~

- This thread signifies to the Doctor thread that the patient is available to be serviced by the Doctor
- initialized to 0

Doctor_ready_nurse ~

- This semaphore signals to the nurse thread that a particular doctor with a given ID is ready.
- This is an array of semaphores each element contains a semaphore that maps to a particular doctor
- The size of this array is number of doctors
- Each semaphore is initialized to 1

Nurse_Finished ~

- This semaphore signals the Patient thread that its operations have been completed and allows the patient thread to continue on to its next set of introduction.

- This is an array of semaphores each element contains a semaphore that maps to a particular nurse
- The size of this array is number of doctors
- Each semaphore is initialized to 1

Doctor_Finised ~

- This semaphore signals the patient that the doctor services to that particular patient have been completed allowing for the patient thread to continue on
- This is an array of semaphores each element contains a semaphore that maps to a particular doctor
- The size of this array is number of doctors
- Each semaphore is initialized to 1

Queue_shield ~

- enforces mutual exclusion in the nurse, receptionist line
- Initial value = 1

Map_Shield ~

- enforces mutual exclusion in the nurseTodoctorMap
- Initial value = 1

Pseudo Code Design :

Class Patient

```
int threadNum;
int DoctorNum

Void run {
receptionist_register.wait()

queue_shield.wait()

Receptionist_line.push()

queue_shield.signal()

patient_ready_receptionist.signal()

Recectionist_Finished[threadNum].wait()

print()

nurse_ready.wait()

queue_shield.wait()
Nurse_line.push()
queue_shield.signal()

patient_ready_Nurse.signal()

Nurse_Finished[threadNum].wait( )

doctor_ready.wait()

Doctor_ready_nurse[DoctorNum].wait()

map_shield.wait()
nurseTodoctorMap.put()
map_shield.signal()
print()

patient_ready_Doctor.signal()

Doctor_Finished[this.threadNum].wait()

print()
print()
}
```

Class Receptionist

```
int Patient_id
Run ()
{
patient_ready_receptionist.wait()

queue_shield.wait()
Patient_id = Reception_line.pop()
queue_shield.signal()

Receptionist_Finisehd[Patient_id].signal()

receptionist_register.wait()
}
```

Class Nurse

```
threadNum;
Local_patient;
Run ()
{
patient_ready_Nurse.wait()

queue_shield.wait()
Local_patient = Nurse_line.pop()
queue_shield.signal()

local_patient.setDoctorNum(threadNum)

print()

Nurse_Finished[local_pratinet.threadNumber].signal();


nurse_ready.signal()

}
```

Class Doctor

```
int threadNum

Patient patient_obj;

run()
{

patient_ready_Doctor.wait()

map_shield.wait()
Patient_obj = nurseTodocorMap.remove(threadNum)
map_shield.signal()


patient_ready_Doctor.wait()


map_shield.acquire();
patient_obj = Main.nurseTodoctorMap.remove(this.threadNum);
map_shield.release();

patient_ready_Doctor.release();
if(patient_obj ==null)
{
patient_ready_Doctor.release();

}
Else
{
print ()


Doctor_Finished[patient_obj.getThreadNum()].release();

doctor_ready.release(); // let new patients enter

Doctor_ready_nurse[this.threadNum].release(); // let new patients enter the correct thread
}
}

Main ()

{
    Num_patients ← user input
    Num_Doctor ← user input

          for(int i = 0; i < num_patients; i++)
          {
                    patient _obj = new Patient(i)
                    Patient_obj.start();
                    patientArrayList.add(patient_obj)
          }

          for(int i = 0; i num_Doctors; i++)
          {
                    Doctor_obj = new Doctor(i);
                    Doctor_obj.start();

                    Nurse_obj = new Nurse(i);
                    Nurse_obj.start();
          }

          Receptionist_thread.start()


          for(int i = 0; i < num_patients; i++)
          {

                    patientArrayList.get(i).join()
          }

          System.exit(0);


}
```