```
// Robby Hallock
// Group D
// robert.hallock@okstate.edu
// 4/26/2022

#include "Corey.h"
#include "Kyle.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>
#include <pthread.h>
#include <sys/types.h>
#include <time.h>

/**
 * Patient tries goes to the clinic for a checkup, but leaves if its full
 * struct threadStruct *contents is a pointer to a stuct containing the patient information
 */

void patientThreadFunc(struct threadStruct *contents)
{
   contents->threadID = (int)gettid(); // gets thread id

   // arrives at clinic
   printf("Patient %d (Thread ID: %d) Arrived to clinic\n", contents->id, contents->threadID);

   // checks if the waiting room is full
   pthread_mutex_lock(&mutex[0]);
```

```
    if (totalRoomCapacity < 1) {
        leaveClinic(contents, 0); // leaves if it is
        pthread_mutex_unlock(&mutex[0]);
        return;
    }
    contents->waitTime = clock();
    enterWaitingRoom(); // otherwise, enters waiting room
    pthread_mutex_unlock(&mutex[0]);



    pthread_mutex_lock(&mutex[1]);


    // checks if there if the sofa is full
    if (totalSofaCapacity <= 0) {
        // stands if it is
        printf("Patient %d (ThreadID: %d): Standing in the waiting room\n", contents->id, contents-
>threadID);
    }
    pthread_mutex_unlock(&mutex[1]);


    // sits on sofa when there is room
    while(1){
        pthread_mutex_lock(&mutex[1]);
        if(totalSofaCapacity > 0) {
            sitOnSofa(contents);
            pthread_mutex_unlock(&mutex[1]);
            break;
        }
        pthread_mutex_unlock(&mutex[1]);
```

```
    }

    sem_post(&count[0]); // signal doctores that patient is ready
    sem_wait(&count[1]); // waits for doctor
    getMedicalCheckup(contents); //gets medical checkup

    makePayment(contents); // makes payment
    leaveClinic(contents, 1); // leaves after successful checkup
}

/**
 * staff checks up patients as they come in
 * struct threadStruct *contents is a pointer to a struct containing staff information
 **/

void staffThreadFunc(struct threadStruct *contents)
{
    contents->threadID = (int)gettid();
    while(1){ // loops
        contents->waitTime = clock(); //start----
        pthread_mutex_lock(&mutex[1]);
        if (totalSofaCapacity == maxSofaCapacity) waitForPatients(contents); // waits for patients if non are
ready
        pthread_mutex_unlock(&mutex[1]);
        sem_post(&count[1]); // signals patient
        sem_wait(&count[0]); // waits for patient signal
        if (left >= maxPatients) return; // leave if all patients are done
        contents->waitTime = clock() - contents->waitTime; //end----
        pthread_mutex_lock(&mutex[10]);
```

```
        summary.medicalProAvgWaitTime += contents->waitTime; // add to total medical wait time
        pthread_mutex_unlock(&mutex[10]);
        performMedicalCheckup(contents); // perform checkup on patient
        acceptPayment(contents); // accpets payment from patient
    }
}
/**
 * Patient enters waiting room
 **/

void enterWaitingRoom()
{
    totalRoomCapacity--;
}

/**
 * patient sits on sofa
 * struct threadStruct *contents is a pointer to a struct containing the patient information
 **/

void sitOnSofa(struct threadStruct *contents)
{
    totalSofaCapacity--;
    printf("Patient %d (Thread ID: %d): Sitting on a sofa in the waiting room\n", contents->id, contents->threadID);
}

/**
 * places a task into the queue
```

```
 * struct task task is a struct containing the task information
 **/

void queueTask(struct task task){
   pthread_mutex_lock(&mutex[11]);
   queue[remainingTasks++] = task;
   pthread_mutex_unlock(&mutex[11]);
}

/**
 * takes a task out of the queue
 * return struct task is the task that it took out of the queue
 **/

struct task dequeue(){
   struct task task = queue[0];
   for (int i = 0; i < remainingTasks-1; i++){
      queue[i] = queue[i+1];
   }
   remainingTasks--;
   return task;
}

/**
 * Continously checks the queue for a new task and executes it
 * args is a void pointer because its a thread but unused
 * return is a void pointer because its a thread but unused
 **/
```

```
void* mainThreadLoop(void* args){
   struct task task;
   int flag = 1;
   while(flag){ // loops until flag is false
      pthread_mutex_lock(&mutex[11]);
      if (remainingTasks < 1){ // checks if a task is available
         pthread_mutex_unlock(&mutex[11]);
         continue;
      }
      task = dequeue(); // gets task
      pthread_mutex_unlock(&mutex[11]);
      // selects which function to perform
      switch(task.selector){
         case 0:
            staffThreadFunc(task.args); // staff
            break;
         case 1:
            patientThreadFunc(task.args); // patient
            break;
         case 2:
            flag = 0; // stop thread
            break;
      }
   }
}
```