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

#ifndef ROBBY_H
#define ROBBY_H

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


// global variables
int successfulCheckups;
int avgStaffWaitTime;
int patientsLeft;
int avgPatientsWaitTime;
pthread_mutex_t mutex[12];
sem_t count[2];
int totalRoomCapacity;
int totalSofaCapacity;
int maxSofaCapacity;
```

```
int checkupTime;
int left;
int buffer;
int maxPatients;
struct summary summary;

struct threadStruct
{
   char *occupation;
   int id;
   int threadID;
   int bondId;
   clock_t waitTime;
};

// struct for tasks
struct task
{
   int selector; // selects which funtion to perform
   struct threadStruct *args; // struct for function arguments
};

struct task queue[256];
int remainingTasks;
/**
 * 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*);
```

```
/**
 * staff checks up patients as they come in
 * struct threadStruct *contents is a pointer to a struct containing staff information
 **/
void staffThreadFunc(struct threadStruct*);
/**
 * Patient enters waiting room
 **/
void enterWaitingRoom();
/**
 * patient sits on sofa
 * struct threadStruct *contents is a pointer to a struct containing the patient information
 **/
void sitOnSofa();

/**
 * 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);
/**
 * places a task into the queue
 * struct task task is a struct containing the task information
 **/
void queueTask(struct task task);
/**
 * takes a task out of the queue
 * return struct task is the task that it took out of the queue
```

```
 **/
struct task dequeue();
#endif
```