The second project is supposed to emulate the doctor’s office using threads and semaphores to synchronize the threads. The simulation caps the number of doctors and patients to 3 doctors and 30 patients. In addition, the project will only use semaphores to synchronize the threads.

For the main function until the nurseHelp function, programming was simple and easy to complete besides remembering how to use threads and semaphores. A small problem I had was incrementing the index of the customer within finding the index for the patient. I solved that easily by placing the increment after retrieving the patient information. Then realizing that the patients must come in at random so quickly changed it to a struct. I think the place where I struggled the most was calling a doctor that had the same ID as the nurse and to have the same patient. At first, I thought of having a separate semaphore that would block all the doctors that did not have the same ID as the nurse. However, that only continued the problem. Next, I learned about the array of semaphores that will unlock the doctor ID that is the same as the nurse. For certain scenarios, the program executed as instructed, however, if the doctor has the wrong id or another doctor is unlocked or the nurse was initialized with another threadID then the problem of starvation would happen again. I tried to make the doctor thread to continue until the CPU would have a clock interrupt to switch to another thread but that also did not solve the problem. I researched and found out that at times when a thread is initialized and then another thread is initialized right afterwards then the first threads value can be over written. Therefore, I add a struct that will maintain the threadID of each nurse and doctor thread and would not be overwritten and used it for the patient threads as well. That solved my issues.

I have learned how to effectively use threads and semaphores to emulate a real-life scenario. In addition, I learned how to synchronize threads to ensure an order of execution while protecting shared resources between the threads. I also learned that threads could be overwritten by other threads before sharing the resources in the function and how to overcome that situation.

The results of my program is that it would accurately simulate a doctors office with at most thirty patients, three doctors, same number of nurses as doctors, and a receptionist.