Matthew Choi

Project 2 – Apartment Simulator

CS1550

The program aptsim.c utilizes semaphores in order to achieve synchronization. This synchronization is created by creating multiple processes for agents and tenants, and furthermore implementing certain rules that determine the order they execute. The following sections will present why my project is fair, starvation-free, and deadlock-free in that order:

* Fair
  + The tenants that are inserted into the that must wait are placed into a waiting queue by the semaphore. This waiting queue will continue to build as more tenants come into the program. I have a function implemented called **signal()** that takes all of these tenant processes in the order they arrived, and will wake them from sleeping. This is fair, as the process that came in first waited the longest. By freeing this tenant first, it is the fairest implementation.
* Starvation-Free
  + For this prompt, I am assuming there are at most 10 times more tenants than agents. As each agent can serve up to 10 tenants, the program (by definition of itself) would only be starvation-free if there are enough agents in the program. If this is achieved, every tenant would receive a turn within the apartment. After the first 10 tenants have received their turn, all the waiting tenants will then be served by the **next agent** as seen in the viewApt() function. Every new agent apartment opening in openApt() gives these waiting tenants a chance into the apartment, so they never wait forever (starve).
* Deadlock-free
  + My implementation is deadlock free as I have explicit ordering for the mutex semaphore and the order of which processes must wait. For example, I enforce **no sleeping** (with the sleep() function) when the mutex semaphore is locked. Also, a process will never wait when acquiring a lock on a semaphore while it holds the mutex. This prevents an agent from holding mutex and waiting for a tenant, while the tenant is waiting for mutex to give the agent a signal on the agent’s wait. This is how deadlocks are prevented in my implementation.