Allison Thompson

CPSC 380

Hansen

4/15/19

Project 2

For clear organization, I create the Person class for Fred and Wilma’s names and upward velocities, which could be inputted into the Seesaw class for results. I also created a Semaphore class, as Visual Studio does not come with a semaphore import. I used a mutex lock and condition variable to control the semaphore, as well as an int count. This is a counting semaphore, though is used like a binary semaphore in my simulation.

I used 2 threads thrA and thrB and 2 semaphores semA and SemB for my project. Prior to the initialization of the threads in init(), the time of each distance by velocity is calculated. Threads thrA and thrB are public members of the Seesaw class so they may be accessed by the main thread. Thread thrA calls fred(), and thread thrB calls wilma() with these inputted times. After the initialization, thrA and thrB are joined in main() located in CPSC380Project2.cpp.

The semaphores semA and semB are initialized in init() with semA at 1 and semB at 0. Within the function fred() connected to thrA, semA.wait() is prior to each output and semB.signal() is after each output. Within function wilma() connected to thrB, semB.wait() is prior to each output and semA.signal() is after each output, second sleep, and tick iteration. This ensures that Fred and Wilma’s heights will be calculated simultaneously but not attempt to output or iterate at the same time. Because semA is initialized at 1 and semB is initialized at 0, this ensures thrA will be first when alternating with semB to output distance.

