Aaron Sanchez

UIN: 228004762

CSCE 410 P5

Design

To start, the best way to create a FIFO scheduler was to use a data structure that makes first in first out easy, so I created a basic queue data structure. To do this, I used nodes of a linked list, so that I could add and remove in O(1) time and not have to shift the array over every time I wanted to remove from the front of the list. After this was done, it was very easy to implement the scheduler, as all we needed to do was to pop the first thread, which is the current thread, if we are yielding, then push the thread if we are adding it or resuming it. In order to terminate, we simply check if we are using the thread that wants to be removed, if so we delete that thread, if not we simply go through the ready queue and check if the queue is in there. If the thread is inside of the ready queue, then we go through and find which node has it, remove that node from the queue, and delete the thread. Once we go through and have to edit Thread.C, all we really need to do is find a way to link the scheduler to the threads, so we will use an extern, which in my interpretation takes a global variable that has already been assigned a value, and uses that, so that for thread_shutdown we can simply call our scheduler terminate.