Project 1

CSCE 4600

Members: Robert Martinez, Isaac Thomas, Thomas Truong, Clint Wyatt

**Project 1 implementation**

All threads use a while loop to cause the program to run indefinitely to allow the threads to execute multiple times in any order. The first semaphore in a thread is a counting semaphore on one of the lists to ensure that there is a node to unlink followed by a binary semaphore shared by all threads for mutual exclusion.

For thread 1, known as the producer, uses a counting semaphore for the free list because the goal of thread 1 is to move a block from the free list to list 1. The key to making thread 1 to work is having the counting semaphore equal to the number of blocks in the free list – 1 (n -1). The reason for this is the free list must always have at least 1 node. Thread 2 always takes a block from the free list and if there are no blocks in the free list, the program will crash because list 1 in thread 2 will try to access memory that is not there.

Thread 2 uses list 1 as its counting semaphore since the number of blocks from list 1 reduces by 1. While thread 2 unlinks a block from the free list, the number of blocks does not change, so it does not need to use a counting semaphore for the free list and the binary semaphore ensures that no other thread modifies it.

Thread 3, known as the consumer, uses list 2 for the counting semaphore as is the similar to thread 1 just with different lists. However, unlike thread 1, thread 3 adds back to the free list counting semaphore since that resource is being linked back to free list.