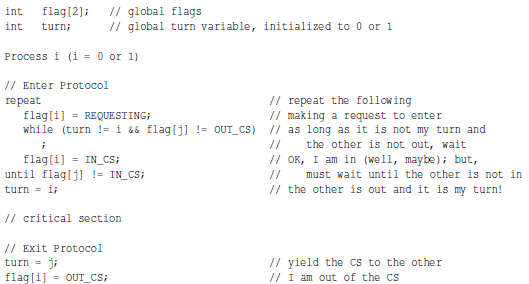
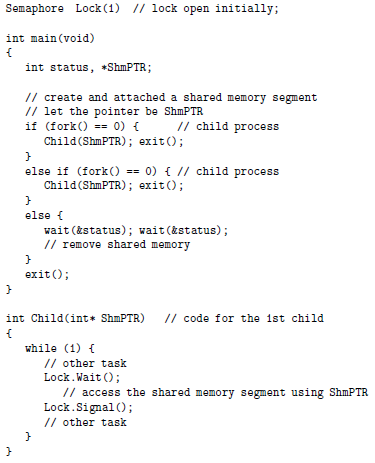
Exam 2 Spring 2015

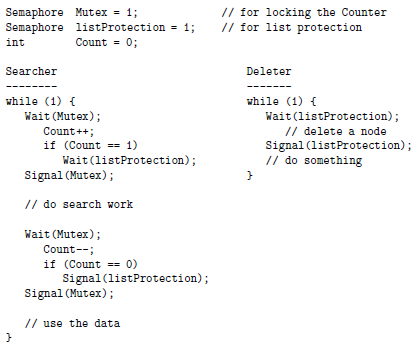
1. Consider the following solution to the mutual exclusion problem for two processes. Prove the solution satisfies mutual exclusion.



1. Define the meaning of a race condition. Use an execution sequence.
2. The semaphore methods wait() and signal() must be atomic to ensure a correct implementation of mutual exclusion. Use execution sequences to show if wait() is not atomic then mutual exclusion can’t be maintained.
3. A programmer wrote a program in which child processes are created and communicate using a shared memory segment. This programmer uses the semaphore capability of ThreadMentor to avoid potential race conditions as follows. However, even though the initialization, process creation, and shred memory segments are correct, this program can never run properly. Identify the problem as clear as possible and provide a convince explanation. Use execution sequences if needed.



1. For the dining philosophers problem, suppose all five chopsticks are placed in a tray at the center of the table and any philosopher can pick up any chopstick. Now, all philosophers must pick up his first chopstick followed by his second chopstick. Is deadlock possible?
2. For the dining philosophers problem, suppose each chopstick is assigned a number. Each philosopher must pick up a chopstick of lower number, then of higher number after he has successfully obtained one. For example, philosopher 3 picks up chopstick C3 and C4. Is deadlock possible?
3. For the dining philosophers problem, suppose an additional chopstick is placed at the center of the table. A philosopher picks up his left chopstick then competes to grab the chopstick at the center. Is deadlock possible?
4. For the dining philosophers problem, suppose an additional chopstick is placed at the center of the table. A philosopher picks up his left chopstick and then competes to grab the chopstick at the center. If the chopstick at the center of the table has been taken, the philosopher tries to grab his right chopstick. Is deadlock possible?
5. Three kinds of threads share access to a singly-linked list: searchers, inserters, and deleters. Searchers examine the list, and can execute concurrently with each other. Inserters append new nodes to the end of the list and must be mutually exclusive. However, one insertion can proceed in parallel with any number of searches. Deleters remove nodes from anywhere in the list. At most one deleter can access the list at a time, and deletion must be mutually exclusive with searches and insertions. Searchers and deleters are the readers and writers in the reader-writer problem, with code below. Write the code for the inserter and add semaphores and variables as needed. Do not modify searcher and deleter. Provide an elaboration to show correctness.



1. A barber shop has one barber, one chair, and n chairs for waiting customers. The barber and customer activities are as follows. The barber is simulated by a thread. When he comes to work, he calls Barber(). The customers are simulated by dynamically created threads. When they need a haircut, they call Customer(). Write the code for these functions are add semaphores and variables as needed.
   1. The barber waits for customers
   2. If there is a customer, the barber brings a waiting customer to the chair and cuts his hair
   3. After serving a customer, the barber sleeps
   4. If there are no waiting customers, the barber sleeps
   5. When a customer arrives, he looks if there is a free chair in the waiting room
   6. If there is a free chair, the customer tells the barber there is a new customer, and sits and waits
   7. If there isn’t a free chair, the customer leaves

