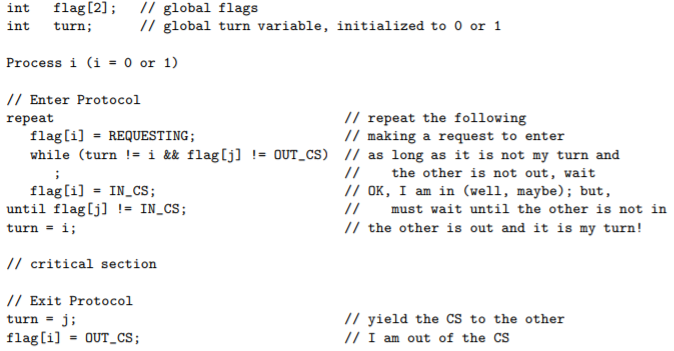
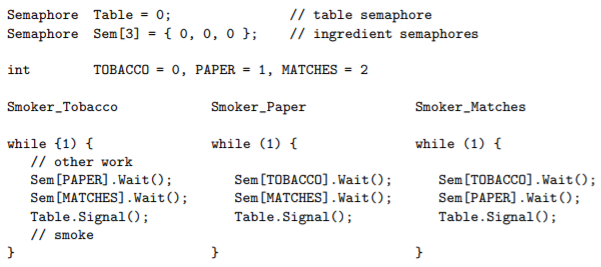
Exam 2 Spring 2014

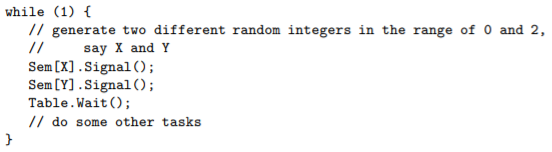
1. Consider the following solution to the mutual exclusion problem for two processes. A process can be making a request REQUESTING, executing in the cs IN\_CS, or have nothing to do with the cs OUT\_CS. This status information, which is represented by an int is saved in flag[i] for process Pi. variable turn is initialized elsewhere to 0 or 1. Note that flag[] and turn are global shared variables. Prove that this solution satisfies mutual exclusion.



1. Define the meaning of a race condition. Use execution sequences.
2. The semaphore methods wait and signal must be atomic to ensure a correct implementation of mutual exclusion. Use execution sequences to show that if wait is not atomic then mutual exclusion can’t be maintained.
3. Three ingredients are needed to make a cigarette: tobacco, paper, and matches. An agent has an infinite supply of all three. Each of the three smokers has an infinite supply of one ingredient only. The following solution uses three semaphores, each of which represents and ingredient, and a fourth to control the table. A smoker waits for the needed ingredients on the corresponding semaphores, signals the table semaphore to tell the agent that the table has been cleared, and smokes for a while.

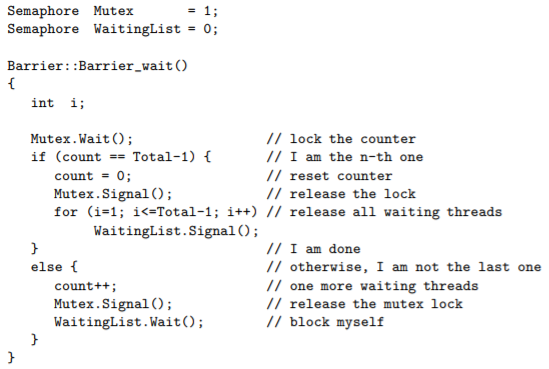


The agent adds two randomly selected different ingredients on the table and signals the corresponding semaphores. Show using execution sequences that this solution can have a deadlock.

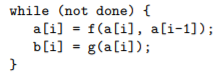


1. A programmer used two semaphores to design a class Barrier, a constructor, and method Barrier\_wait() that fulfills the following:
   1. The constructor Barrier(int n) takes a positive integer argument n and initializes a private int variable in the Barrier class to have the value of n
   2. Method Barrier\_wait(void) takes no arguments. A thread that calls Barrier\_wait() blocks if the number of threads being blocked is less than n-1, where n is the initialization value that will not change during execution. Then the nth calling thread releases all n-1 blocked threads and all n threads continue.

This programmer came up with the following solution. However, he found his solution reacted strangely because sometimes the same thread may be released multiple times in the same batch. Help this programmer pinpoint the program with an execution sequence and explanation.



1. A multithreaded program has two global arrays, a and b, and a number of threads that execute concurrently. Thread Ti runs the following code. Declare semaphores with initial values and add wait and signal calls to thread Ti so that it will compute the results correctly.



1. A unisex bathroom is shared by men and women. A man or woman may be using the room, waiting to use the room, or doing something else. They work, use the bathroom, and come back to work. The rule of using the bathroom is simple: there must never be a man and a woman in the room at the same time, however people of the same gender can use the room at the same time. Declare semaphores and other variables and add wait and signal calls to the threads to that the man threads and woman threads will run properly.

