Gripon, John Adrian G.

BSIT – 301

Part I: Counting Semaphore

Questions:

1. In your perspective, what makes counting semaphore primitives a good concurrency mechanism?

- A counting semaphore can perform the initialize, increment, and decrement operations. It's worth can be found across an unrestricted domain. It regulates access to a resource with numerous instances.

2. How does the structure of counting semaphore primitives differ from binary semaphore primitives?

- The values of the count in semaphore counting are not restricted to just 1 and 0. It regulates access to a resource that is used by several instances.

Part II: Binary Semaphore

3. Briefly explain the purpose of the semWaitB and semSignalB functions in Figure 2.

- The process is sent to semWaitB, which examines its value. If the value is 1, it will decrease until it reaches 0, at which point it will be added to the queue. The process will be halted and loop until the next process is initiated if the value is 0. The process is delivered to semSignalB by semWaitB. The process will be added to the ready list and be available for execution right away if its value is 0. When a process' value is 1, it will be eliminated from the queue until its value is 0.

4. Based on Figures 1 and 2, which semaphore structure is easier to implement and why?

- Compared to the counting semaphore, the binary semaphore is simpler to implement. The only integer values for a binary semaphore are 1 and 0. It is simpler to manage and deploy since it only permits one process at a time to enter the crucial area.

Part III: Monitor

5. Deduce at least one (1) characteristic of a monitor based on Figure 3. Elaborate on your answer.

- A monitor is an instance of a class that may be used safely by several threads, making the monitor mechanism a higher-level mechanism. A monitor executes all of its methods using mutual exclusion. As a result, a method of the monitor can be executed concurrently by one

thread. It is simple to collaborate with the monitor and create the method content of the monitor thanks to this mutual exclusion strategy.

6. Would you agree that a monitor, as a concurrency mechanism, can support process synchronization? Why or why not?

- I agree that process synchronization can be supported via a monitor. The synchronization code in the monitor is centralized in one place, and its users are not required to understand how it works. The code is independent of the quantity of processes; it supports as many processes as you like.