**Wong Kai Yuan (DC026157) CISC2005 Lab 7**

**Question 1**:

1.1 Please complete the code in example1.c.Text

Description automatically generated

1.2 How does the semaphore “s” change as the program runs?

The semaphore “s” initially is 1, when it runs function increment, sem\_wait(s) will decrease “s” by 1, hence “s” become 0. If other thread wish to run the function of increment, it will be blocked and wait until “s” become 1 (once first thread finish the function increment)

So, “s” is 1, then 0, then 1, then 0 …….

**Question 2**:

2.1 Please complete the code in example2.c.

Text

Description automatically generated

2.2 How does the semaphore “s” change as the program runs?

“s” initially is 2 (2 resources, only 2 worker). In the program, it will change between 0-2, and finally finish at “s” back to 2.

“s” is 2. When a thread run “get\_service”, s will decrease by 1, the customer is getting service. Now “s” is 1, if another thread run “get\_service”, same thing happen. “s” is 0 now. If another thread run “get\_service”, because s is now 0, this thread have to wait until the previous (one of) the 2 threads finished and give back the resources.

**Question 3**:

Text

Description automatically generated

Why does the deadlock occur in example 3?

Deadlock will occur in above code structure, this is because first thread is waiting for second, and second thread is waiting for first.

When the first thread acquires semaphore A, then second acquires semaphore B. The first thread then waits for semaphore B to be released, which can only happen when the second thread releases semaphore B. However, the second thread is waiting for semaphore A to be released, which can only happen when the first thread releases semaphore A. Thus, both threads are stuck waiting for the other to release a semaphore, resulting in a deadlock.

**Question 4 :**

Graphical user interface, text

Description automatically generated