**CSE 381 Lab 6**

**1 Objective**

1) Practice C++ programming

2) Practice using synchronization variables to solve problems

3) Develop deep understanding of deadlock problems

**2 Dining Philosophers Problem**

We have discussed the classic synchronization problem in class. Here is the problem statement.

“The Dining Philosophers problem is a classic synchronization problem that illustrates the challenge of deadlock. There is a round table with n plates and n chopsticks arranged as illustrated in Figure 1. A philosopher sitting at each plate requires two chopsticks to eat. Suppose that each philosopher proceeds by grabbing the chopstick on the left, grabbing the chopstick on the right, eating, and then replacing both chopsticks.”

dining.pdf

Figure 1 Dining Philosophers

In this lab, we assume there are 7 phosphors and only 7 chopsticks. Implement this problem using C++. Download lab6.tar from Canvas.

**2.1 Design Chopstick Class (10 points)**

In this problem you are required to design the chopstick class.

This class has two public methods: void get( ) and void put( ).

If the chopstick is available, the get( ) method sets the chopstick to be used and returns. Otherwise, the calling thread is blocked.

For the put( ), it somehow sets the chopstick be available and wakes up the waiting thread on this chopstick.

In the C++, it is a good practice to put the class declaration and the class implementation to

header file and implementation file respectively. In addition, the initialization work can be done inside the class constructor method.

**Turn-in** Chopstick.h and Chopstick.cc files

**2.2 Implement Dinging Philosophers(DP) (30 points)**

The DP problem has 7 philosophers and 7 chopsticks. You are required to use the class Chopstick and the philosopher thread to model the chopstick and philosopher. In order to help you debug your code, print the message “Philosopher i pciks up the chopstick j” and “Philosopher i puts down the chopstick j” for these events. Please implement it in the dp.cc.

Hint:

1. Create an array Chopsticks[ ] with seven elements.
2. Create 7 philosopher threads and each thread has an index: user\_id. You can use Chopsticks[user\_id ] and Chopsticks[(user\_id +1)%7] to represent the philosopher user\_id’s left and right chopstick.

Complie:

Download Makefile from Canvas and issue command:

make dp

**Turn in** chopstick.h, chopstic.cc and dp.cc.