# CS4532 Concurrent Programming

Take Home Lab 2

**Due – Nov 26 before 11:55 PM**

## Learning Outcomes

In this lab we will learn how to use mutexes and semaphores in an actual program. At the end of the lab you will be able to:

- understand how to devise a solution to a problem that requires synchronization
- develop a program to solve a synchronization problem using a well-known programming language

## Challenge

This problem was originally based on the Senate bus at Wellesley College. Riders come to a bus stop and wait for a bus. When the bus arrives, all the waiting riders invoke *boardBus*, but anyone who arrives while the bus is boarding has to wait for the next bus. The capacity of the bus is 50 people; if there are more than 50 people waiting, some will have to wait for the next bus. When all the waiting riders have boarded, the bus can invoke *depart*. If the bus arrives when there are no riders, it should depart immediately.

### Task

Write a program with appropriate synchronization code that enforces all of these constraints in Java, Python, or C#.

### Notes

- This problem is taken from the book "Little book of Semaphores", page 211. It is ok to look at the solution. But if you have the same answer and you cannot explain how it works, you will not get any marks (randomly selected set of students will be invited for a one-on-one grading session).
- The reason for forcing you to use Java is that implementing synchronization code in Java is relatively easier than several other languages
- Note that busses and riders will continue to arrive throughout the day. Assume inter-arrival time of busses and riders are exponentially distributed with a mean of 20 min and 30 sec, respectively.
- Each lab group consists of up to 2 students. You are free to select your lab buddy; however, it should not be the same buddy that you had for Lab 1 or 2.
- Students are encouraged to talk to the lecturer and instructors for any clarifications. Discussions on Moodle are encouraged too.

**What to Submit**

- Submit following files as a single .zip file
  - Source files
  - README.txt explaining how to run your program
  - Screenshot of output from the program
- Name the .zip file as **lab3_*<index no 1>*_*<index no 2>*.zip**. Replace *<index no x>* with your index number.