15-418/618 Spring 2021
Exercise 7

Assigned:	Fri., Apr. 2
Due:	Fri., Apr. 9, 11:00 pm

Overview

This exercise is designed to help you better understand the lecture material and be prepared for the style of questions you will get on the exams. The questions are designed to have simple answers. Any explanation you provide can be brief—at most 3 sentences. You should work on this on your own, since that's how things will be when you take an exam.

You will submit an electronic version of this assignment to Canvas as a PDF file. For those of you familiar with the LaTeX text formatter, you can download the template and configuation files at:

http://www.cs.cmu.edu/~418/exercises/ex7.tex
http://www.cs.cmu.edu/~418/exercises/config-ex7.tex

Instructions for how to use this template are included as comments in the file. Otherwise, you can use this PDF document as your starting point. You can either: 1) electronically modify the PDF, or 2) print it out, write your answers by hand, and scan it. In any case, we expect your solution to follow the formatting of this document.

Problem 1: Lock Implementations

For the following questions, we will compare different aspects of three lock implementations: test-and-test-and-set, ticket-based, and array-based lock.

A. Releasing a lock requires a write. Order the three locks based on their interconnection traffic caused by the release. Briefly justify your ordering.

B. Compare the space requirements for the three lock implementations.

C. What is the advantage provided by the ticket-based and array-based locks versus a test-and-test-and-set (with or without exponential backoff)?

Problem 2: Transactional Memory

The following code sequence attempts to use transactional memory to update a location in memory, and falls back on a lock when conflicts are detected; however, it has a flaw where updates can be lost. Please explain the flaw and how the code can be updated to prevent it. Recall that `*loc += val` is three instructions in x86 assembly.

```
void atomic_add(int* loc, int val)
{
  int result = xbegin();
  if (result == SUCCESS)
  {
    *loc += val;
    xend();
  }
  else
  {
    lock();
    *loc += val;
    unlock();
  }
}
```