# Assignment 4

CS 494 - Principles of Concurrent Programming - S20

## Description

This assignment has four options: Undergraduate (1, 2, and 3) and Graduate (4). The options are described below:

### Option 1, 2, and 3

This option requires you to resubmit Assignment 1, 2, or 3 (respectively). You have to start from the implementation you submitted and improve it, using the feedback you received from the instructors. You are encouraged to take code from the solutions given out by the instructions to fix your submission.

Undergraduate students have to choose the option respective of the assignment with the lowest grade. For instance, if you scored 80%, 90%, and 100% in Assignment 1, 2, and 3 (respectively), you have to choose Option 1. If two assignments have the same lowest grade, you can choose from either.

**Empty submissions will result in keeping the lowest grade from Assignments 1, 2, and 3 as the grade of Assignment 4.**

Graduate students **cannot** use Options 1, 2, and 3.

### Option 4

This option requires you to implement your own ReadWriteLock and use it in your implementation of Assignment 2. Your implementation should be fair, as described in class and in the textbook.

This option reuses the code and the tests for Assignment 2. If your submission did not use ReadWriteLocks as described in the solution released by the instructors, you have to start from the code in the instructor's solution.

Undergraduate students can choose this option in alternative to the lowest grade in the Options 1, 2, or 3 above.

Graduate students **have to** choose this option.

## Due Date and Late Policy

- This assignment is due on **May 2** (Saturday) by **5pm CST**.
- Submissions delivered by **May 3** (Sunday) by **5pm CST** will have a 10% penalty.
- Submissions delivered by **May 4** (Monday) by **5pm CST** will have a 25% penalty.
- **No submission will be accepted after May 4 past 5pm CST (Monday)**

The code and date used for your submission is defined by the last commit to your git repository.

## Submission and Grading

This assignment should be submitted through Github, and has an automatic grade component of **80%** for Options 1 and 3; and **70%** for Options 2 and 4. You can check your current grade at any point by submitting your code and checking Travis. The automatic grade is determined by tests, that will check if your project outputs the expected result. Each test is worth 10%.

This assignment also has a question component, which should be completed on **Gradescope**. There are two questions about your implementation, worth either 20% or 30% depending on the option you use:

**Option 1, 3**: (20%) How did you improve your original submission?

**Option 2**: (30%) How did you improve your original submission?

**Option 4**:

1. (10%) How did you ensure that no reader can overtake a writer waiting for the lock?
2. (10%) In your implementation, can one writer overtake another writer that was waiting for the lock for longer? Why not or how can this happen?

## Bonus Points

This assignment has a total of **10% bonus points**, which you can earn by using Piazza as described in the syllabus. Your posts should be: public, tagged with the **Assignment 4** label, and non-anonymous to the instructors to count towards the bonus.

## Errors and Omissions

If you find an error or an omission, please post it on Piazza as soon as you find it.

## Hardcoding and Academic Integrity

Any hardcoding will result in a 0% grade. Hardcoding is when you submit code that detects which test is being run, and simply outputs the expected result. For instance, detecting that test 22 is running, and replacing the usual execution of your submission with `System.out.println("expected result")`.

The academic integrity policy described in the syllabus applies to this assignment. You are responsible for writing all the code that you submit. We will use an automatic tool that detects plagiarism on all submitted code, and we will investigate all instances where plagiarism is more than likely.

Please refer to the syllabus for the full academic integrity policy.