UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems

Practical 3

- **Date issued:** 11 August 2023
- **Deadline:** 18 August 2023 (Midnight)
- This practical consists of 1 task. Read **carefully!**

1 Introduction

1.1 Objectives and Outcomes

This practical aims to explore foundations of concurrent shared memory computing. That is, a shared-memory computation consists of multiple threads, each of which is a sequential program in its own right. These threads communicate by calling methods of objects that reside in a shared memory.

You must complete this assignment individually. Copying will not be tolerated.

1.2 Submission and Demo Bookings

While other practicals may contain starting code for you to build on, for the purposes of this practical, all code must be written from scratch.

Submit your code to **clickup** before the deadline.

You will have to demonstrate each task of this practical during the **physical** practical lab session. Booking slots will be made available for the practical demo.

1.3 Mark Allocation

For this practical, in order to achieve any marks, the following must hold:

- Your code must produce console output. (As this is not marked by fitchfork, formatting is not that strict)

- Your code must not contain any errors. (No exceptions must be thrown)
- Your code may not use any external libraries for **locking** apart from those already provided.
- You must be able to explain your code to the tutors and answer any questions asked.

The mark allocation is as follows:

Task	**Marks**
Implementation	5
Explanation	5
Total	10

2 Practical Requirements

Chapter 4 extensively covers foundations of concurrent shared memory computing using registers. The registers are not intended to be a practical model for computation. Instead, the textbook prefer easy-to-understand but inefficient constructions over complicated but efficient ones.

You are tasked to demonstrate the following:

- One can **construct** a wait-free MRMW Atomic register **from** SRSW Atomic registers.

2.1 Implementation

The following must be completed:

- Threads, readers and writers, to access the shared memory a number of times
  - Readers: Read values via the read method and output, thread-name and returned value.
  - Writers: Write random value via write method and output, thread-name and written value.
- MRMW atomic register, which would require the implementation of the following classes:
  - MRMW atomic
  - MRSW atomic
  - SRSW atomic
  - StampedValue
  - Register

2.2 Note

- This is a lock-free implementation. Most of the code is provided in the textbook/slides.
- Understand the following transitions, from:
  - SRSW to MRSW then MRMW (Number of Readers and Writers)
  - Safe to Regular then Atomic (Degree of consistency)
  - Boolean to M-Valued (Range of values)