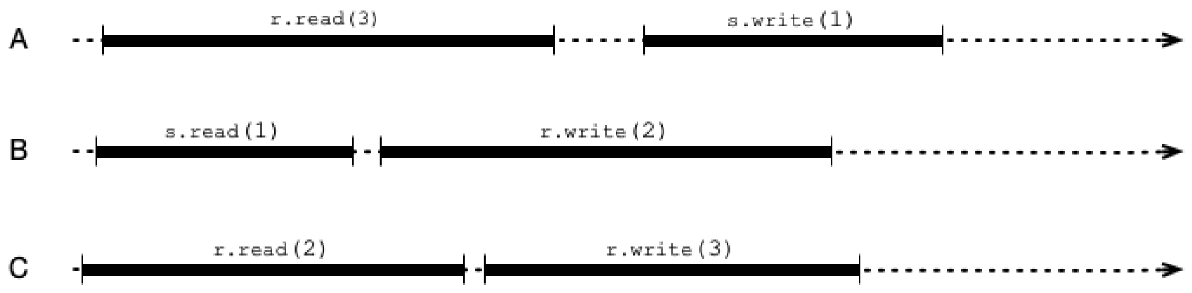
Homework 03

COSC 273: Parallel and Distributed Computing, Spring 2023

Due: Friday, 04/14/2023 at 11:59 pm

Instructions: You may work on this assignment in groups of up to 3 and submit a single solution for your group. All group members are responsible for understanding all submitted solutions.

Exercise 1. Consider the following histories of executions of read/write registers (variables), `r` and `s`. Note that the *result* of the `read` operations are written as arguments so that, for example, `r.read(3)` means that a process read `3` as the value of `r`.



Please explain your answers to the following questions.

(1) Restricting attention *only* to register `r`, is the execution sequentially consistent? Linearizable?

(2) Restricting attention *only* to register `s`, is the execution sequentially consistent? Linearizable?

(3) Is the entire execution (including both registers) sequentially consistent? Linearizable?

Exercise 2. Consider the following queue implementation, `IQueue`. For simplicity, assume that the array `items` is unbounded.

```
public class IQueue<T> {
    AtomicInteger head = new AtomicInteger(0);
    AtomicInteger tail = new AtomicInteger(0);
    T[] items = new (T[]) Object[Integer.MAX_VALUE];

    public void enq(T x) {
        int slot;
        do {
            slot = tail.get();
        } while (!tail.compareAndSet(slot, slot+1));
        items[slot] = x;
    }

    public T deq() throws EmptyException {
        int value;
        int slot;
        do {
            slot = head.get();
            value = items[slot];
            if (value == null)
                throw new EmptyException();
        } while (!head.compareAndSet(slot, slot+1));
        return value;
    }
}
```

(a) Describe an execution demonstrating that `IQueue` is *not* linearizable.

(b) Is `IQueue` Lock-free? Wait-free? Why or why not?