Homework 03

Instructions

You may work in groups of up to 4 and submit a single assignment for the group. For computational problems, please show your work; for conceptual questions, please explain your reasoning. Solutions may be neatly hand-written and scanned or typeset. Please submit your solution to Moodle **in PDF format**.

Due: Friday, April 2, 23:59 AoE

The following exercises refer to the queue implementation, `IQueue`. For simplicity, assume that the array `items` is unbounded. You can read about the `compareAndSet` method for the `AtomicInteger` class here.

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

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

    public int 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;
```

```
    }
}
```

Exercises

Exercise 1. Give an execution demonstrating that `IQueue` is *not* linearizable.

Exercise 2. Is `IQueue` wait-free? Lock-free? Why or why not?