A Countdown Latch

Due Date:

This assignment is due Monday, 2017-01-30 *at the beginning of lecture*.

Purpose

The purpose of this lab is to practise designing synchronization objects using threads and channels as primitives.

Synchronization Barrier Description

(See also the documentation for the JAVA class `java.util.concurrent.CountDownLatch`.) Sometimes a number of threads need to wait around until they are all ready to go on to the next activity. A synchronization barrier, or count-down latch is a synchronization object to help with this. It contains an internal non-negative integer counter, set when the latch is constructed, and has two atomic methods:

- `(latch-wait latch)` which causes the calling thread to wait until the internal counter reaches zero; and
- `(latch-decrement latch)` which decrements the internal count. If the latch count reaches zero, then *all* threads waiting on the latch resume execution.

A count-down latch is not re-usable. Once its count reaches zero, both `latch-wait` and `latch-decrement` have no effect.

Programming

⇒ Implement in RACKET, or give a detailed design for a count-down latch.

⇒ Implement in RACKET, or give a detailed design for tests for a count-down latch.