Goals for this Lab

Implement Parallel Spanning Tree construction using:

- isolated
- Java Atomics

Measure scalability on NOTS.

isolated

Recall from lecture:

- `isolated(() -> { S1; });`
  - No other isolated sections execute in parallel (global isolation)
- `isolated(obj1, obj2, ..., () -> { S2; });`
  - No other object-based isolated sections execute in parallel for which there is a non-empty intersection of the object set
  - Global isolation == object-based isolation on all objects in program

In addition:

- `isolatedWithReturn(obj1, () -> { ...; return S3; });`
  - Global or object-based isolation with a return value

Atomics

Recall from lecture:

- AtomicInteger
  - Supports atomic updates of an integer value by any number of threads
  - https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/atomic/AtomicInteger.html

In addition:

- AtomicReference
  - Supports atomic updates to a stored object reference
  - https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/atomic/AtomicReference.html

AtomicReference example

```
// reference to null
Boolean ref = null;

finish (() -> {
    async (() -> {
        ref = Boolean.TRUE;
    });

    async (() -> {
        ref = Boolean.FALSE;
    });
});

System.out.println("Referring to " + ref.get());
```

AtomicReference example

```
// reference to null
AtomicReference<Boolean> ref = new AtomicReference<Boolean>();

finish (() -> {
    async (() -> {
        ref.compareAndSet(null, Boolean.TRUE);
    });

    async (() -> {
        ref.compareAndSet(null, Boolean.FALSE);
    });
});

System.out.println("Referring to " + ref.get());
```

Use AtomicReference to safely update ref.

Equivalent example with global isolation

```
// reference to null
Boolean ref = null;

finish (() -> {
    async (() -> {
        isolated(() -> {
            ref = Boolean.TRUE;
        });
    });

    async (() -> {
        isolated(() -> {
            ref = Boolean.TRUE;
        });
    });
});
```

Spanning Tree

Will be using isolated and Atomics to parallelize spanning tree construction.

See Lecture 20 and provided sequential implementation for more details on spanning trees and their parallel construction.

Submitting

Show your passing tests on NOTS, through the autograder or manually.

Commit your changes back to your turnin folder.