CS 511: Homework Assignment 1
Due: September 9, 11:55pm

1. Assignment Objectives

Get acquainted with the notions of

- State diagram
- Trace
- Interleaving
- Implementation of threads in Java

2. Assignment Policies

Collaboration Policy. Homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me.

3. Assignment

Exercise 1

Consider the program:

```
global int n = 0;

thread P: {          thread Q: {
  while (n < 2)        n = n + 1;
    print(n);          n = n + 1;
}                    }
```

Note: in answering the following exercises you may assume that assignment is atomic.

1. Draw the state diagram for this program.
2. Supply the execution traces that print the following sequences: 012, 002, 02.

3. Should 2 necessarily appear in the output?
4. How many times can 2 appear in the output?
5. How many times can 1 appear in the output?
6. How many times can 0 appear in the output?
7. What is the length of the shortest sequence of outputted numbers that can be exhibited?

Exercise 2

Implement a class `AssignmentOne` that includes a method (further details of this class will be supplied later)

```
public List<Integer> lprimes(List<Integer[]> intervals)
```

that given a list of arrays of integers of size 2 $[[g_1, d_1], \ldots, [g_k, d_k]]$ such that

- the list $[g_1, d_1, \ldots, g_k, d_k]$ is increasing and
- each number is greater or equal to 2

creates k threads where each thread i computes the list of primes in the interval $[g_i, d_{i+1})$ (notice the interval is semi-open). For example, given the list of arrays

$$[[1, 101], [101, 201], [201, 301], [301, 401], [401, 501]]$$

it will spawn 5 processes. Process 2 will, for example, compute all the primes between 101 and 200, namely

[101,103,107,109,113,127,131,137,139,149,151,157,163,167,
173,179,181,191,193,197,199]

The result of the whole process would be the list

[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101,
103,107,109,113,127,131,137,139,149,151,157,163,167,173,179,181,191,193,197,
199,211,223,227,229,233,239,241,251,257,263,269,271,277,281,283,293,307,311,
313,317,331,337,347,349,353,359,367,373,379,383,389,397,401,409,419,421,431,
433,439,443,449,457,461,463,467,479,487,491,499]

Additional guidelines:

1. The list of arrays should be built from arguments that are passed on to `main`. For example, the list $[[1, 101], [101, 201], [201, 301], [301, 401], [401, 501]]$ arises from the user having passed the arguments 1 101 201 301 401 501 to the `main` method of class `AssignmentOne`.
2. The class `Assignment` should make use of a helper class `PrimeFinder` that implements the `Runnable` interface and has the following attributes:

```
private Integer start;
private Integer end;
private List<Integer> primes;
```

and the following methods:

```
PrimeFinder(Integer startNum, Integer endNum) // Constructs a PrimeFinder

public List<Integer> getPrimesList() // Returns the value of the attribute primes

public Boolean isPrime(int n) // Determines whether its argument is prime or not

public void run() // Adds all primes in [this.start,this.end) to the attribute primes
```

3. In `AssignmentOne` you will require a list of threads. Also, you will have to wait for all the threads in the list to finish before collecting the results, which should be placed in an attribute called `results` of class `AssignmentOne`. In order to wait for a thread `t` to finish you may use the command `t.join()`.

4. Submission Instructions

Submit a zip file named `traces_<Surname>.zip` (where `<Surname>` should be replaced by your surname) through Canvas containing:

1. a file named `traces_<Surname>.pdf` with the answers to the exercises, and
2. the files `PrimeFinder.java` and `AssignmentOne.java` with the implementation of exercise 2.

Important: Please make sure that your assignment compiles and runs on the Linux lab before submitting.