CSCE 314 [Sections 202, 502] Programming Languages – Fall 2016
Anandi Dutta
Assignment 7
Assigned on Friday, December 2, 2016

Electronic submission to eCampus due at 23:59, Sunday, 12/11/2016
By electronically submitting this assignment to eCampus by logging in to your account, you are signing electronically on the following Aggie Honor Code: "On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment."

Note 1: This homework set is individual homework, not a team-based effort. Discussion of the concept is encouraged, but actual write-up of the solutions must be done individually.

Note 2: Turn in one yourLastName-yourFirstName-hw5.tar or yourLastName-yourFirstName-hw5.zip file on eCampus, nothing else. What to submit is detailed below.

Note 3: All Java code that you submit must compile without errors using javac of Java 8 (that is installed in the departmental servers, linux.cse.tamu.edu and compute.cse.tamu.edu). If your code does not compile, you will likely receive zero points for this assignment.

Note 4: Remember to put the head comment in your files, including your name, your UIN, and acknowledgements of any help received in doing this assignment. Again, remember the honor code.

Note 5: Problem [1], [2] and [3] contains 30 points each. If you write a detailed design document for problem [4] describing your implementation, you will get 10 points. If you solve the Problem [4], you will get 20 points as extra credit. Use the piazza forum to ask any question related this HW.

[1] We have provided an example of ParallelMergeSort.java file in Piazza that uses the join/fork Framework in Java. Therefore, you have the structure/flow of how to divide a problem into independent sub-problems and finally join them and form the final result. Moreover, please check out the Piazza, we have posted a file named helpHW6 under HW section. It contains some helper code, you can use them in your solution if you wish.

Revise the ParallelMergeSort.java file and define a generic parallelMergeSort method as follows:

```
public static <E extends Comparable<E>> void parallelMergeSort(E[] list)
```

[2] Implement the following method in parallel to sort a list using quick sort

```
public static void parallelQuickSort(int[] list)
```

Write a test program that times the execution time for a list of size 9,000, 000 using this parallel method and a sequential method.

[3] Suppose you have multiple processors, so you can speed up the matrix multiplication. Implement the following method in parallel.

```
public static double[][] parallelMultiplyMatrix( double[][] a, double[][] b)
```

Write a test program that measures the execution time for multiplying two 2,000 * 2,000 matrics using the parallel method and sequential methød, respectively.

[4] You are working as a programmer for developing a game show. The host of the show asked you to develop a Java program that will help the game show running smoothly. The rules for the show is that it will ask the player a couple of questions. If the player can answer the question correctly in 10 seconds, he/she will score one/1 point. If the player's answer is wrong or if he/she does not answer the question within 10 seconds, the correct answer is displayed and the player does not score any points. After getting an answer or running out of time, the next question is displayed.

From the rules above you can get the sense this program needs to run in parallel manner to handle different tasks at the same time. You can use semaphore/locks to ensure thread synchronization. We suggest that you should use three threads to handle three tasks. Below is a skeleton of logic that you might follow:

```
printQuestion thread{

// start timer thread
// wait for answer or timeout
// save response from user
// print "correct", "wrong" or "timeout"
}


reader_thread{

      // check for end of program
      // read answer;
     // move answer to global variable
     // release printer thread
}

timer_thread(int num) {
      // wait 10 seconds; you can use sleep() here
      // still waiting for this question?
      // put TIMEOUT value as response
      // release printer thread
      }
```

Again, it is not necessary that you have to follow this skeleton code. If you have better/different idea/implementation, you are welcome to follow that. Moreover, add a test in main to interact with the user.