COMP 322: Fundamentals of Parallel Programming (Spring 2017)
Instructors: Vivek Sarkar, Mack Joyner
Worksheet 1: due at end of class today

Name: _____ Netid: _____

Honor Code Policy for Worksheets: You are free to discuss all aspects of in-class worksheets with your other classmates, the teaching assistants and the professor during the class. You can work in a group and write down the solution that you obtained as a group. If you work on the worksheet outside of class (e.g., due to an absence), then it must be entirely your individual effort, without discussion with any other students. If you use any material from external sources, you must provide proper attribution.

1) Parallelizing your weekday/weekend tasks!

Consider the sequential list of weekday/weekend tasks below. Assume that you have an unbounded number of helpers to help you with your chores and tasks. *Insert async and finish pseudocode annotations to maximize parallelism, while ensuring that the parallel version has no unintended/undesirable outcomes.* Make any reasonable assumptions e.g., you only have one fridge, you have access to multiple washers & dryers, you can reorder statements so long as you don't change the outcome, etc.

```
Watch COMP 322 video for topic 1.2 by 1pm on Wednesday

Watch COMP 322 video for topic 1.3 by 1pm on Wednesday

Make your bed

Clean out your fridge

Buy food supplies and store them in fridge

// Run two loads of laundry
{

     Run load 1 in washer

     Run load 2 in washer

     Run load 1 in dryer

     Run load 2 in dryer

}

Call your family

Post on Facebook that you're done with all your tasks!
```

2) Parallelizing Matrix Multiply

Consider the sequential version of a matrix-multiply algorithm shown below that computes the product of two NxN matrices A and B into an NxN matrix C, assuming that all entries in C were initialized to zeros. (Matrices are represented as 2D arrays in Java.)

Insert async and finish pseudocode annotations to maximize parallelism, while ensuring that the parallel version always computes the same result as the sequential version. Pay attention to the scoping of the async and finish constructs.

```
for (int i = 0 ; i < N ; i++) {
  for (int j = 0 ; j < N ; j++) {
    for (int k = 0 ; k < N ; k++) {
      C[i][j] = C[i][j] + A[i][k] * B[k][j];
    } // for-k
  } // for-j
} // for-i
. . . // Print matrix C
```