CSEP 524: Assignment #2

(due prior to class, Tuesday January 22nd)

1) Reading:

a) Lin & Snyder, Ch 3 (pp. 61-85)
b) Lin & Snyder, Ch 6, POSIX Threads section thru Perf. Issues (pp. 145-173)

Submit 1-2 discussion questions per chapter for consideration in class discussions.

2) **Chapel Task Parallelism:** Using the framework code provided on the course webpage (treeSearch.chpl), modify the provided sequential tree search routine to use task parallel features, making it a parallel tree search. Read the comments in the code file for more details and step-by-step instructions. No performance timings are required for this exercise (though we'll be curious to hear what you find if you do any), just the prescribed code modifications.

3) **Embarassingly Parallel Performance Study (continued):**

a) Install Chapel if you haven't already and make sure it works (see class webpage).

b) Complete problem 4b from assignment 1.

c) Also complete problem 5 for Chapel; however, for brevity you only need to implement and run the (ramp, factorial) case for the two distributions. Vary numTasks and compare the results to that for C+Pthreads.

Notes:

- Once you have your Chapel program working correctly, remember to compile it with --fast before doing performance runs (to turn off the runtime checks).

- If comparing Chapel vs. C+Pthreads performance, be sure that you are using the same size integers (most C compilers default to 32-bits while Chapel defaults to 64-bits).

- Chapel has some built in timing capabilities provided in its standard 'Time' module. Add 'use Time;' to the top of your program to import its symbols into your program. See examples/primers/timers.chpl for a brief introduction.