CSEP 524: Assignment #7

(due prior to class, Tuesday March 5th)

1) **Reading:**
   a) *A Brief Overview of Chapel*, Sections 9.3.4-9.4 (pp. 15-18)

   Submit 1 question ***in a textfile format*** for consideration in class discussions by Monday evening, 9pm, Feb 25th .

2) **Multi-Locale Smith-Waterman in Chapel:**

   a) Starting from the file sw-framework.chpl, implement the Smith-Waterman algorithm for distributed memory (multi-locale) execution in Chapel using the following high-level strategy:
      - The provided code reads in the two sequences in from input files specified by configs seq1file and seq2file;
      - Distribute the seq1len x seq2len domain & matrix into a vertical panel per locale using the Block distribution;
      - Create a task per locale;
      - When safe/signaled, have each task compute 'rowsPerChunk' (set via a config) rows of its vertical panel at a time, serially;
      - Use a Block-distributed (seq1len by rowsPerChunk) x numLocales array of sync vars to have each locale signal to the next when it can start working on its corresponding chunk.

      Note that the provided framework provides a lot of code for you – the input, the serial computation of a submatrix, the backtracing, and the output of the difference between the sequences. It also supports a completely serial The goal is for you to focus on the parts related to the multi-locale parallel execution: specifying the distributions, task parallelism, and synchronization required to have the execution fully pipelined once locale #0 has computed numLocales-1 chunks. A series of TODO comments guide you to the places in the code that require modification.

      Important Note: The framework is designed to support both a local/nondistributed and parallel/distributed version of the algorithm at any given time, controlled via the config param 'computeInParallel'. Be sure to set this to true (via the compiler flag -scomputeInParallel=true or by modifying the source code) when you're ready to start compiling and running your parallel framework.

   b) In a paragraph or two of text, describe how the MPI implementation of the algorithm in part (a) would be implemented. What communications would you use? What aspects of the implementation would be more difficult? What aspects would be easier?

c) Starting with your answer from part (a), create a second version of the program that takes a similar, but slightly different approach: Rather than creating all of the tasks *a priori*, when each task finishes a chunk, have it fire off a task on the next locale to compute the same set of rows once it is legal to do so. For grading purposes, please do this in the file sw-part2.c and comment your changes w.r.t. part (a) using comments that start with "PART C: ".

[*Goal:* get some experience with Chapel in a multi-locale setting; get some experience with wavefront algorithms in distributed memory settings; compare and contrast global-view PGAS-style programming with SPMD message passing]

Optional: See if you can obtain speedup with this approach on longer sequences using a distributed memory architecture like the UW cluster.

Optional: Use tasking within each locale to parallelize the computeChunkSerially() routine and take full advantage of the cores available to you.

Optional: Do something else cool with multilocale parallelism in Chapel.

3) **Think of Final Lecture Questions** One of my goals for my final lecture slot is to use it as a "requests night" and come back around to any brushed-past topics or unanswered questions—e.g., things you've wondered about from the class and never gotten a chance to ask or discussion questions that you asked but we never got around to answering. This would be a good week to start thinking about what you would personally like to see covered in that slot.

4) **That's it!** If you finish early, get a jumpstart on your final project.