Programming Assignment #4 **Due April 11**
Introduction to Parallel Processing MTH/CSE 4082

Write code (C/C++ or FORTRAN) to test the Collatz conjecture for all integers from up to **Nmax**, recording the highest integer, **High**, ever reached in the process. See the slides "Week 9 Assignment 4 and HOTPO" for information. Use openMP to parallelize the outer loop over the integers from **1** to **Nmax.**

Run the code on blueshark with a gnu openMP compiler. Your task is to investigate the speedups you can obtain using different scheduling. Test with at least static, dynamic, and guided scheduling of the for loop with differing chunk_size. You may explore other options as well, google "OpenMP loop scheduling".

Excellent research questions to guide your investigation are:
1) What's the max speedup I can get on this hardware?
2) Does it depend on Nmax?
3) Does it depend on how the for loop is scheduled?

Turn in

- A listing of your code
- A table of **High** for **Nmax**=2000,20000,200000,2000000
- Write up including a full description of the tests you ran and your conclusions

Helpful suggestions

- The reduction clause (slide 21 of "Week 6 Shared Memory Intro B") supports the max operation in OpenMP versions 3.1+.

  See http://www.techdarting.com/2013/06/openmp-min-max-reduction-code.html

- Since we want to look at large integers, **long** or better yet **long long** are appropriate choices for the integer variables declarations. Note that OpenMP for loops can use either of these, but not the unsigned variants.

- I don't expect you to find a counterexample to the Collatz conjecture, but you still should put a bound on the number of times the inner HOTPO iteration loop is executed. I suggest using **Imax**=**Nmax**.

- I'm being deliberately vague in the instructions. I'm not telling you exactly what size problem and number of threads to use. Hopefully you've learned a bit about parallel computing issues this term, and can design an interesting speedup experiment of your own. ***That said, if you are unclear on how to proceed, please ask.***

- You should be able to see some speedups. I do. If the runtime of your code seems independent of the number of threads, its likely that your problem is too small, or your code is not really running in parallel.

- For your *.sh file to submit jobs on blueshark, see the OpenMP examples on canvas. The only thing you change to change the number of threads is `--cpus-per-task=1` to, say `--cpus-per-task=4`