CS 380P Parallel Systems — Spring 2013	Programming Assignment #1
Parallel Prefix in Pthreads	Due January 23, 2013

In this assignment, you will use Pthreads to implement a generalized parallel prefix framework. The primary goal is to give you experience in using Pthreads on the TACC cluster, so you may work in pairs, and **this assignment will not count much.** You may also consult with other students—even those outside your pair—on this assignment, but your team of two should write your own code, and you should acknowledge any help that you have received.

In addition to learning some basics of Pthreads, we hope that this assignment will give you an appreciation for the power of the parallel prefix operation, so you will use your framework to solve two specific problems. The textbook provides several examples of such problems on pages 117 and 118 (compute team standings, etc), but we encourage you to show some creativity and solve some other problems with this framework. **We will provide extra credit for particularly creative uses of this framework.**

You should produce a written report that describes the problem that you solved, your approach to solving this problem (which for this assignment can be quite brief, since you can cite the textbook), your method of testing your code, any tricky or interesting aspects of your effort, and any insights that you have.

1 Details

You may write your Pthreads code in either C or C++, and you should run it on the TACC Lonestar cluster (`lonestar.tacc.utexas.edu`), particularly if you wish to time your program (which we strongly encourage). This cluster runs Linux and provides compute nodes with 8 Intel Nehalem processors and 48GB of RAM (sixteen of the nodes have 144GB of RAM). Each node also has 2 nVidia GPUs, but for this assignment you will not need to use these. You can login to this machine through ssh using your TACC username and password. If you have trouble accessing this machine, please email the TA with your name and TACC username.

The webpage `https://portal.tacc.utexas.edu/group/tup/user-guides/lonestar` provides additional info about setting up your environment (shell, etc.), writing and building programs on Linux, and so on.

2 Hints

Before you get started, be sure to read the notes that the TA has posted on our Piazza page, which contains all sorts of useful details and hints.

Figures 5.6 and 5.7 of the textbook provide sketches of your code. The code given in Figure 5.7 of the first printing is *incomplete*; see `http://www.cs.utexas.edu/users/lin/revfig/Fig5-7Corrected.pdf` for a correct version. For background on the Peril-L notation, read pages 88-100 (Chapter 4) of the textbook. Chapter 6 of the book provides background on Pthreads. For more details on Pthreads, visit the following web sites:

```
https://computing.llnl.gov/tutorials/pthreads/
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
```

We strongly recommend that you develop and debug your programs locally (on departmental or other linux systems) before moving on to Lonestar. You should be able to run POSIX threads on any of the departmental workstations, albeit with no true parallelism. If you need a UTCS account, please visit the following web site, and note that it typically takes a day or two to get your account:

`https://apps.cs.utexas.edu/udb/newaccount/`

Lonestar is a batch system with possibly uncertain queue lengths. Please submit your jobs well in advance of the deadline. For best results in this class, we recommend that you use the Development Queue, which places restrictions on the running time of your program and the number of processors used in exchange for faster turnaround time. Again, see the TA's web page for more details.

As with any problem, we suggest that you start with a simple problem, perhaps prefix sum, before you move on to other problems. We also suggest that you experiment with different number of threads and different data set sizes.

What to Turn In

- Explain your work by writing a brief statement that states what you did and how you did it. This is also your chance to point out your clever tricks, your assumptions, and any difficulties that you encountered. Don't forget to acknowledge any collaborators.
- Explain your testing methodology. How confident are you that you have no race conditions or bugs?
- Provide some evidence that your program is correct by providing sample program output.
- Optionally document the performance of your program's execution on `lonestar`.
- Turn in your brief statement, your documented program listing, any timing results, and any other additional exposition that you deem relevant.

3 Late policy

Your assignment is due at **11:59pm** on the due date. Late submissions will be penalized 10% per day.