Parallel Programming **Assignment 2**
FS 2011 Out: 2011-03-03
Department of Computer Science Due: 2011-03-10
ETH Zurich

Parallel Sieve of Eratosthenes

In this exercise you will look into potential for concurrency in the Sieve of Eratosthenes and will have to implement a parallel version of it.
The result should be a class $Primes$ in C# that implements the $IEnumerable < long >$ and allows to iterate over prime numbers that are less or equal than a certain maximum max (e.g. 200000000), where max should be an argument for the constructor. Since you are developing a parallel algorithm, the second argument should define the number of threads that are used for computations. It should be possible to not specify the number of threads, in which case the number of cores in the system should be taken.
Recall the algorithm to find prime numbers using the sieve:

1. Create a list of natural numbers 2, 3, 4, 5, ..., max. None of which is marked.
2. Set k to 2, the first unmarked number on the list.
3. Repeat:
   (a) Mark all multiples of k between k^2 and max.
   (b) Find the smallest number greater than k that is unmarked. Set k to this new value. Until $k^2 > max$.
4. The unmarked numbers are primes.

Use the following idea for parallelization of the algorithm:

1. Using Eratosthenes, compute primes up to $\sqrt{max}$ and store them in an array a.
2. Build p chunks of roughly equal length covering the range from $\sqrt{max} + 1$ up to max, where p is a number of threads.
3. Create a thread for each chunk (except for the first one) and fork it. Use the main thread to process the first chunk. Pass a reference to a to all threads.
4. Each thread has to use the (read only) array a to get the "seeds" k to mark the numbers in its own chunk.
5. The main thread after processing the first chunk waits for all threads to join.

Obviously with this approach up to $\sqrt{max}$ computations will be done sequentially. Before you start experiments use Amdahl's law to predict the speed up. Write down your predictions.

The solution should contain your implementation of $Primes$ class and some other class with the Main method that tests your implementation. This method should also measure time that takes to initialize an instance of $Primes$ (i.e. compute the prime numbers). Compute it for different number of threads starting from 1 and up to 4x number of cores in your computer. Compare your experimental results with your predictions. Explain results. Include the measurements into the report.

What to turn in

Please turn in your solution into your SVN directory. Please place your explanations and predictions in either a text file or in the source code as a comment.
Please commit source files only. **Do not** commit: dll, exe, pdb etc.