CS 590V: Parallel Computing
Spring 1997: Midterm 2
April 16, 1997, 10:30 - 11:20 AM

1. **[30 points]** Consider a functional parallel formulation of the sieve. A list of numbers `a[0..255]` with `a[i] = i` passes through two stations; the first eliminates all multiples of 2 and the second, all multiples of 3 (in a complete sieve, there will be other stations as well, but for the sake of simplicity, we only want to eliminate multiples of 2 and 3). Associate a thread with each of these. Note that the multiples of 3 can be eliminated from a segment of the array only after multiples of 2 have been eliminated. Break the array into 32 parts of 8 elements each. The first thread eliminates multiples of 2 from the first 8 elements and passes them to thread 2 and so on. Assume that when an element is eliminated, its value is changed to -1. Write a thread-based pseudocode for this simplified problem (with two threads). Make sure you take care of all locks and synchronization.

2. **[20 points]** Consider a scenario in which processors are organized in a ring (with a wraparound after the last processor). Processor P_i can send work to processor P_j iff $j > i$. In this case, a simple ring termination detection mechanism works as follows:

   - Processor P_0 initiates a token after it finishes its local work and sends it to processor P_1. This token can be in the form of a single integer.
   - Processor P_i after finishing its work waits for the token from processor P_{i-1} and sends it to processor $P_{(i+1) \text{ modulo } p}$; where p is the number of processors.
   - When the token returns to processor P_0, we can infer that all processors are done with the work.

   This is a simplification of the generalized ring termination detection algorithm. Assume that all the work is contained in a function called `work()`. Do not worry about the function itself; just make a call to it in your program. Write a message passing pseudocode for the simple termination detection strategy.